UNIVERSITÉ DU QUÉBEC

MÉMOIRE PRÉSENTÉ À
L'UNIVERSITÉ DU QUÉBEC À TROIS-RIVIÈRES

COMME EXIGENCE PARTIELLE
DE LA MAÎTRISE EN ÉLECTRONIQUE INDUSTRIELLE

PAR
PIERRE BROUARD

L'APPLICATION DU FILTRE DE KALMAN À L'APPROXIMATION DE MESURANDES PAR UNE FONCTION SPLINE - ÉTUDE D'UN ALGORITHME ET SON IMPLANTATION EN DSP.

SEPTEMBRE 1994

Résumé

Le traitement du signal mesuré dans un système de mesure consiste en deux phases: conversion et reconstitution. La conversion dans un système de mesure électrique consiste en une série de transformations de signaux reçus d'un objet de mesure, en des signaux électriques standards, préférablement numériques. Ces derniers sont ensuite mis au traitement afin d'obtenir des estimés des valeurs des grandeurs à mesurer; cette opération est nommée "reconstitution de mesurande".

La reconstitution d'un mesurande consiste en l'estimation d'un signal x, qui n'est pas mesurable directement, à partir des résultats de mesure d'un autre signal $\tilde{y}$, qui est lié avec le premier de façon causale. Les méthodes de reconstitution sont généralement basées sur certaines suppositions. Celles-ci concernent le modèle mathématique de la relation entre les signaux, l'information accessible a priori du signal à reconstruire et le bruit qui entache les résultats de conversion $\tilde{y}$.

Le problème de reconstitution, posé d'une façon abstraite, est bien conforme aux nombreuses situations pratiques dans les différents domaines de la technique de mesure, particulièrement en spectrométrie. Malgré sa simplicité logique ce problème est difficile du point de vue numérique à cause de son conditionnement déficient.

Parmi plusieurs méthodes de régularisation du problème de reconstitution, les méthodes paramétriques et probabilistes sont les plus efficaces. Le projet consiste donc en une étude et un développement d'un algorithme de reconstitution de mesurande basé sur l'application du filtre de Kalman pour l'estimation des paramètres d'une fonction spline supposée approximer le mesurande et aussi en l'implantation de cet algorithme dans un processeur numérique de signaux (DSP) en vue de l'application pour la correction de données spectrométriques.

L'algorithme à étudier est basées sur une paramétrisation du signal x par une fonction spline et sur une méthode probabiliste d'estimation de paramètres de cette fonction, nommée filtre de Kalman. L'application de l'algorithme de reconstitution pour la correction de données spectrométriques exige une certaine vitesse de calcul pour que le

temps de correction soit acceptable pour des utilisateurs du spectromètre.C'est la motivation essentielle de l'utilisation d'un processeur numérique de signaux (DSP).

Les éléments principaux de la méthodologie de la recherche proposée sont l'élaboration et l'étude de l'algorithme de reconstitution en MATLAB et sur DSP ainsi qu'en une synthèse des résultats de l'étude.

Le résultat principale de ce projet de recherche est un algorithme original de reconstitution de mesurande basé sur l'application du filtre de Kalman pour l'estimation des paramètres d'une fonction spline supposée approximer le mesurande implanté dans un processeur numérique de signaux (DSP).

Les études de l'algorithme développé permettent de conclure:

- que l'algorithme donne des résultats de qualité supérieure comparé à d'autres méthodes de reconstitution pour les signaux fortement bruités;
- que l'algorithme est souple et facile à utiliser;
- qu'un précalcul du gain de Kalman doit être fait pour des applications différentes;
- que l'algorithme demande peu de mémoire et que le traitement est très rapide;
- que l'exactitude des résultats de reconstitution obtenus par l'implantation DSP est comparable aux résultats obtenus par MATLAB surtout pour des niveaux de bruit élevés.

Donc, l'algorithme est étudié et il est prêt à être utilisé dans d'autres traveaux de recherche poursuivis au Laboratoire de système de mesure de l'U.Q.T.R.

Remerciements

Je veux profiter de cette occasion pour remercier les gens qui m'ont aidé, appuyé et encouragé pour réaliser ce travail mais aussi tout au long de ma maîtrise.

Premièrement, je remercie mon directeur de maîtrise, Monsieur Andrzej Barwicz, pour m'avoir donné la possibilité de travailler dans son laboratoire avec une bonne équipe et du bon équipement, et donc d'enrichir mes connaissances avec des appareils d'une telle technologie. Je le remercie pour la confiance qu'il m'a accordée et la liberté d'action qu'il m'a permise.

Aussi, je désire montrer ma satisfaction au professeur Roman Morawski pour son appui et l'apport de ses idées lorsque j'ai cru que l'algorithme ne fonctionnerait pas.

Je suis pour toujours reconnaissant à mes parents pour l'aide morale et financière. Merci aussi, Linda, pour la correction des fautes d'orthographe.

J'ai aussi beaucoup apprécié l'aide de Monsieur Daniel Massicotte dans le domaine du filtre de Kalman ainsi que l'aide de Monsieur Mohamed Ben Slima en DSP. Ils étaient au doctorat avec du travail jusque par dessus la tête.

Table des matières

Liste des figures

Liste des Tables

La métrologie est la plus simple et la plus complète solution d'un problème sur lequel n'a cessé de travailler l'esprit humain: exprimer des quantités ou des rapports à variations continues, à l'aide de règles syntaxiques applicables à un système de signes individuels ou discontinus...

Cournot, fond. connaïss., 1851 P. 305.

I. Introduction

L'électricité, voici un vaste domaine. Il se décompose en sous-domaines tels que l'installation, la puissance, le contrôle, etc... Partout, la mesure de phénomènes électriques est nécessaire. Plusieurs autres domaines de science utilisent l'électricité de façon intensive. Dans ces domaines, on mesure des phénomènes physiques. Les résultats des mesures servent à comprendre le système étudié dans un cadre précis. Certains systèmes sont très complexes et demandent des quantités énormes de mesures. Les calculs devant être faits avec ces résultats de mesure, appelés données, échantillons, ou points, peuvent être très lourds et fastidieux. De nos jours, l'ordinateur a remplacé la règle à calculer. Malgré tout, l'entrée de ces quantités spectaculaires de données dans la mémoire de l'ordinateur pourrait être un long travail. Pour éviter cela, des moyens automatiques permettent d'entrer ces données instantanément. Ces moyens demandent par contre que les résultats de mesure du phénomène soient sous forme électrique en format binaire. L'organe permettant ce transfert entre phénomène physique et phénomène électrique se nomme un capteur. Il existe souvent plusieurs types de capteurs pour chaque type de phénomène à mesurer. De plus, chaque capteur crée un signal électrique dont les caractéristiques varient selon la mesure faite. Les caractéristiques électriques peuvent être la tension, la fréquence, la phase, la largeur d'impulsion, etc... Le signal électrique est alors converti en binaire par un convertisseur analogique/numérique et assimilé par l'ordinateur où il sera traité.

Nous sommes maintenant prêts pour une définition très générale de la mesure. D'après L. Finkelstein [1]:

La mesure est un procédé opérationnel d'application des nombres aux éléments d'une certaine classe des aspects ou des caractéristiques de l'univers suivant des règles bien définies.

Les règles sont conçues de façon à ce que le nombre, appliqué à l'entité, la décrive. Donc, une relation entre les nombres, appliquée aux différents éléments de la classe, correspond à une relation empirique entre des éléments auxquels ils sont appliqués.

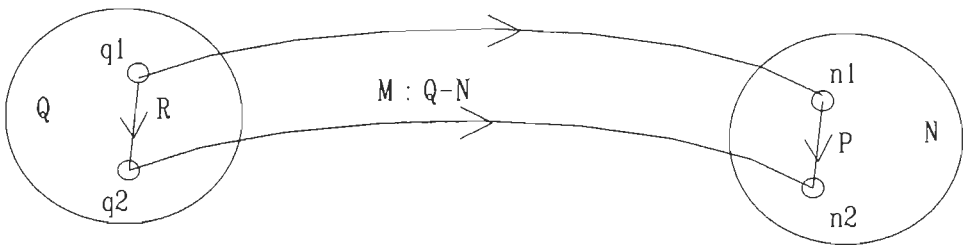


Fig 1: Diagramme de la mesure générale.

Q une classe bien définie des entités extramathématiques (q1.....qn)
R un ensemble des relations empiriques existantes dans la classe (r1.....rn)
N un ensemble de nombres (n1.....nn)
P un ensemble de relations numériques définies dans l'ensemble de nombres (p1.....pn)
M une transformation du domaine Q vers N, c'est un homorphisme d'un système relationnel empirique (Q,R) et d'un système numérique relationnel (N,P)

Par la transformation M (procédé opérationnel bien défini) :

$n_i \in N$ devient l'image de $q_i \in Q$ et il est noté par $n_i=M(q_i)$;
n_i est appelé la mesure de q_i;
q_i est la valeur mesurée;
Q est la classe des valeurs mesurées.

Nous avons déja montré les différentes transformations nécessaires pour effectuer une mesure de façon électrique. Le schéma de la figure 2 montre de façon globale et générale les parties d'un système de mesure. La figure 3, pour sa part, montre un exemple classique d'application électronique d'un tel système.

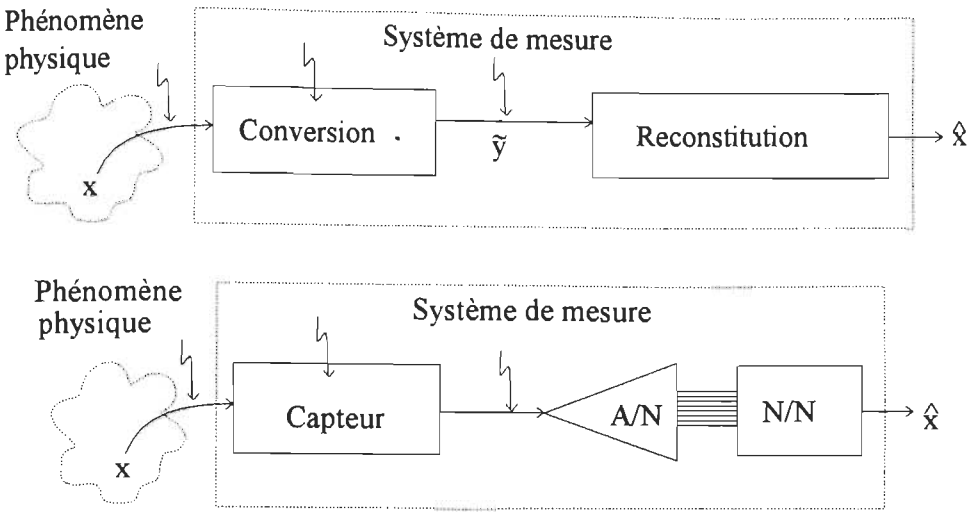


Fig 2 : Système de mesure a) général, b) électronique classique.

Le premier bloc de la figure 2 représente un instrument de mesure qui transforme le phénomène en résultat de mesure brut. Les flèches croches démontrent à quel niveau du système, le bruit peut affecter la mesure. Le résultat brut $\tilde{y}$ est alors reconstitué par un algorithme quelconque pour obtenir une approximation du signal original x soit $\hat{x}$.

L'exemple classique de la figure 3 démontre le système mentionné en début d'introduction. Un capteur fait la conversion du signal quelconque en signal électrique analogue et un convertisseur A/N formate les données afin qu'elles puissent être traitées par un ordinateur. Dans l'ordinateur, (bloc N/N de la figure3) l'algorithme traite le signal brut (donc bruité) pour effectuer l'approximation ($\hat{x}$) du signal d'entrée (x).

Aucun instrument de mesure n'est parfait: donc, lors d'une mesure, le signal à mesurer (le mesurande) est déformé par l'instrument (le système de mesure). De plus, le bruit présent dans tout système physique, s'ajoute aux résultats de mesure. Ces deux phénomènes, la déformation par l'instrument et le bruit additif, peuvent causer de grandes erreurs de mesure. Ces erreurs peuvent être réduites en utilisant les méthodes appropriées de reconstitution du mesurande. Ces dernières consistent en l'approximation du mesurande basée sur le résultat brut de mesure. Très souvent, dans le cas de systèmes stationnaires linéaires, ce résultat peut être modélisé de façon adéquate par la convolution du mesurande et de la réponse impulsionnelle de l'instrument. Mais, en pratique, on ne peut pas simplement appliquer une déconvolution pour obtenir le mesurande car cette opération est extrêmement sensible au bruit. Alors, des méthodes plus sophistiquées doivent être utilisées pour la reconstitution du mesurande.

Ce travail consiste en l'étude et le développement d'un algorithme implanté sur MATLAB pour l'estimation des paramètres d'une fonction spline approximant le mesurande. Cet algorithme est basé sur le filtre de Kalman correspondant au modèle mathématique de la relation entre le mesurande et le résultat brut de mesure sous forme d'un système d'équations d'état; on l'appellera: canal de mesure. Pour appliquer le filtre de Kalman, on suppose que le mesurande est modélisé avec une suite de variables aléatoires (identiques, noncorrelées) dont la variance est connue *a-priori*. On suppose, en plus, que la variance du bruit qui entache le résultat brut de mesure, est aussi donnée. L'efficacité de l'algorithme sera évaluée en utilisant des données synthétiques et réelles de type spectrométrique. De plus, les résultats seront comparés aux résultats obtenus par d'autres méthodes. La validité des résultats s'étend également à d'autres cas de mesure où la relation entrée/sortie est semblable.

L'implantation de l'algorithme dans un processeur DSP 56001 est tentée et discutée. Le but est, bien sûr, de diminuer la taille du système de mesure. L'ordinateur est utile pour le développement de l'algorithme mais il serait intéressant, un jour, d'intégrer le traitement à la conversion dans le même instrument de mesure miniature!

Nous verrons donc en détail ce qu'est la reconstitution et la raison de sa nécessité. Nous expliquerons ensuite la théorie et l'utilité des fonctions spline pour la mesure ainsi que la théorie du filtre de Kalman. Nous pourrons ensuite utiliser ceci pour expliquer l'algorithme à l'étude dans ce mémoire. Nous verrons l'importance des différents paramètres et contraintes. Nous suivrons les étapes et les difficultés rencontrées lors de l'implantation sur Matlab, language C et DSP. Certains des résultats obtenus seront alors présentés et comparés. Nous pourrons alors conclure sur la faisabilité, l'exactitude, l'utilité, l'implantabilité et la viabilité de l'algorithme à l'étude.

II. La reconstitution

Nous avons vu qu'une partie importante d'un système de mesure est la reconstitution du mesurande. Pour certains systèmes de mesure simples, la partie reconstitution est implicite au point de ne pas nous en apercevoir lorsque nous sommes en train de la faire, comme c'est le cas lors d'une mesure de distance avec un étalon physique. Pour des systèmes complexes, nous acceptons parfois l'erreur introduite par l'instrument. Dans ce cas, la reconstitution n'est pas effectuée, comme lors de l'utilisation d'un oscilloscope dans la région approchant la fréquence de coupure. Pour des systèmes de mesure où le résultat doit être précis et où l'instrument déforme le résultat, la reconstitution est alors indispensable.

Pour ce faire, nous devons avoir un modèle représentant la transformation introduite par le capteur. Un capteur linéaire introduira une certaine constante de proportionnalité ou de décalage pour chaque composante fréquentielle. La reconstitution sera alors simple. Pour des mesures de phénomènes complexes, les capteurs apportent des déformations importantes et variables selon la plage de mesure et/ou selon un autre phénomène physique; par exemple le temps, la température, la longueur d'onde. Les données prennent alors la forme d'un vecteur pouvant représenter un signal. Nous comprenons que pour de tels systèmes, le modèle représentant la transformation sera plus complexe.

Dans la théorie des systèmes électriques et en traitement de signal, nous utilisons souvent l'intégrale de convolution pour modéliser le transfert de données. Nous utilisons le schéma bloc de la figure 3 a) pour schématiser cette convertion et la convolution est décrite par:

$$y = \int_{-\infty}^{\infty} g(t-\tau)\cdot x(\tau)d\tau \qquad (1)$$

Pour une fonction de transfert (g) causale, i.e.: existe pour $t \geq 0$ seulement, la discrétisation de (1) donne:

$$y_n = \sum_{i=0}^{n-1} g_{n-i}\cdot x_{i+1} \qquad \text{pour } n = 1, 2, \ldots, N \qquad (2)$$

Cette convolution numérique peut ensuite être mise sous forme matricielle:

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} = \begin{bmatrix} g_1 & 0 & \cdots & 0 \\ g_2 & g_1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ g_N & \cdots & g_2 & g_1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix} \tag{3}$$

ou plus simplement:

$$y = G \cdot x \tag{4}$$

La convolution peut modéliser aussi les systèmes simples selon la fonction g, appelée transopérateur ou réponses impulsionnelles. En effet, prenons l'exemple ou le transopérateur est une impulsion de Dirac centrée à zéro et d'amplitude A. Le résultat de convolution devient alors:

$$y(t) = A\, x(t) \tag{5}$$

une simple constante de proportionnalité.

Ce modèle semble donc assez général mais, bien sûr, demande de connaître la réponse impulsionnelle du capteur. La reconstitution devient donc un problème de déconvolution tel que schématisé à la figure 3 b).

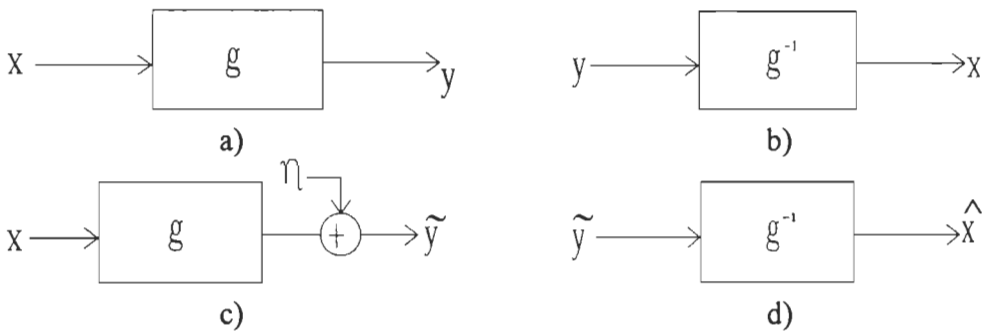


Fig 3 : a) convertion simple, b) reconstitution simple, c) influence du bruit lors de la convertion, d) reconstitution général.

La déconvolution étant l'opérateur qui résout matriciellement, par oposition à (4):

$$x = G^{-1} \cdot y \tag{6}$$

ou

$$x_n = \sum_{i=1}^{n} G^{-1}_{n,i} \cdot y_i \qquad \text{pour } n = 1, 2, \ldots, N \tag{7}$$

Malheureusement, cette opération est très sensible au bruit et ce bruit est incontournable, il existe dans tout système électrique et s'ajoute toujours à un plus ou moins haut niveau, selon l'environnement, aux signaux en traitement. Puisque la déconvolution est sensible à ce bruit, il est important de le considérer dans notre modèle tel que schématisé à la figure 3 c).

Puisque nous ne connaissons pas les valeurs des réalisation du bruit η, nous ne pouvons pas les considérer directement lors de la reconstitution. Nous devons donc résoudre le problème de reconctitution généralisé schématisé à la figure 3 d).

L'exemple suivant illustre la cause d'une telle sensibilité au bruit. Soit:

$$x = \begin{bmatrix} 0 \\ 2 \\ 5 \\ 2 \\ 0 \end{bmatrix} \quad \text{et} \quad g = \begin{bmatrix} 0.01 \\ 1.01 \\ 2.01 \\ 3.01 \\ 0.01 \end{bmatrix} \quad \text{donc} \quad G = \begin{bmatrix} 0.01 & 0 & 0 & 0 & 0 \\ 1.01 & 0.01 & 0 & 0 & 0 \\ 2.01 & 1.01 & 0.01 & 0 & 0 \\ 3.01 & 2.01 & 1.01 & 0.01 & 0 \\ 0.01 & 3.01 & 2.01 & 1.01 & 0.01 \end{bmatrix}$$

Par convolution, selon (4):

$$y = \begin{bmatrix} 0.01(0) + 0 + 0 + 0 + 0 \\ 1.01(0) + 0.01(2) + 0 + 0 + 0 \\ 2.01(0) + 1.01(2) + 0.01(5) + 0 + 0 \\ 3.01(0) + 2.01(2) + 1.01(5) + 0.01(2) + 0 \\ 0.01(0) + 3.01(2) + 2.01(5) + 1.01(2) + 0.01(0) \end{bmatrix} = \begin{bmatrix} 0 \\ 0.02 \\ 2.07 \\ 9.09 \\ 18.09 \end{bmatrix}$$

Maintenant, soit un vecteur de bruit additif de faible niveau et de valeur aléatoire:

$$\eta = \begin{bmatrix} 0.0001 \\ 0.0001 \\ 0 \\ 0 \\ 0.0001 \end{bmatrix}$$

Le vecteur de sortie perturbé est donc:

$$\tilde{y} = y + \eta = \begin{bmatrix} 0.0001 \\ 0.0201 \\ 2.0700 \\ 9.0900 \\ 18.0901 \end{bmatrix}$$

Si on veut reconstituer x à partir de $\tilde{y}$ selon (7), on applique:

$$\hat{x} = G^{-1} \cdot \tilde{y} \qquad \hat{x} = \begin{bmatrix} 0.01 \\ 1.00 \\ 103.99 \\ -9798.00 \\ 970204.01 \end{bmatrix}$$

On voit clairement la différence entre x et $\hat{x}$ lors d'une reconstitution sans régularisation. Comme deuxième exemple, la figure 4 montre le signal d'entrée x(t) et le transopérateur g(t). Leur convolution additionnée d'un bruit normal donne le signal de sortie perturbé $\tilde{y}(t)$. En utilisant $\tilde{y}(t)$ et g(t) pour faire une simple déconvolution, nous obtenons le signal d'entrée reconstitué $\hat{x}(t)$. Cet exemple montre clairement la nécessité d'un bon algorithme de reconstitution puisqu'il est évident que $\hat{x}(t)$ ne ressemble pas à x(t).

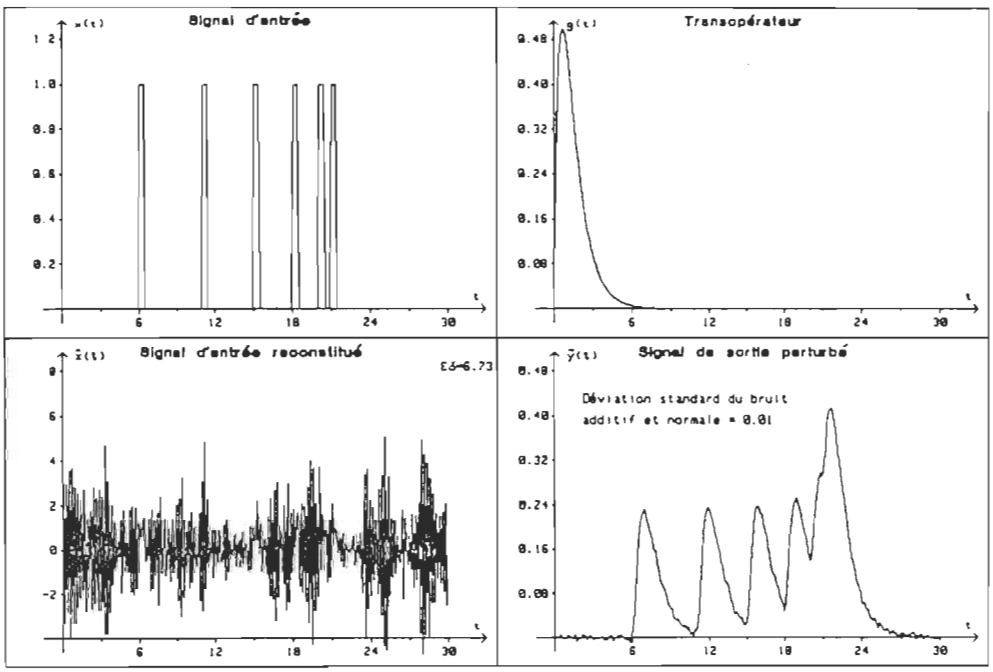


Fig 4: Exemple: effet du bruit lors de la déconvolution simple.

En plus de montrer l'inefficacité de cette méthode de reconstitution, la figure 4 montre bien l'effet du transopérateur sur la sortie. On voit que les pointes ont été atténuées, élargies, biaisées et même superposées pour ce qui est des deux impulsions les plus rapprochées. On comprend maintenant l'intérêt des méthodes de reconstitution plus efficaces.

Plusieurs domaines de recherche scientifique et de technique industrielle utilisent les méthodes spectrométriques, lesquelles représentent un cas particulier où s'applique ce type de modélisation. Plusieurs méthodes existent pour reconstituer le mesurande dans ce cas [1]. Mais on cherche toujours des méthodes nouvelles pour améliorer l'exactitude générale de la reconstitution, la vitesse, l'immunité contre le bruit et la facilité de l'implantation matérielle.

Plusieurs méthodes existent bien sûr pour faire la reconstitution. Les méthodes itératives sont souvent utilisées. La méthode de Van Cittert fut une des premières méthodes inventées. Elle est très simple mais ne convient qu'à très peu de cas. Plus tard, Jansson améliora la méthode de Van Cittert en y ajoutant des contraintes. Ces contraintes diminuent le nombre de solutions possibles et améliorent considérablement les résultats. D'autres méthodes viennent de Gold et Tikhonov, utilisant respectivement le théorème de Bayes et des méthodes spectrales.

Des méthodes basées sur l'approximation du mesurande par une fonction spline sont très intéressantes parce qu'elles peuvent être très précises [2]. L'application de modèles purement discrets demande une interpolation des résultats obtenus entre les échantillons; cette interpolation peut causer des discontinuités des dérivées de la fonction résultante. Par contre, la fonction spline a des dérivées continues. Le principal défaut des méthodes de reconstitution basées sur les fonctions spline est la quantité de mémoire nécessaire pour obtenir tous les paramètres de ces fonctions.

Les méthodes basées sur le filtre de Kalman sont pour leur part simples à implanter et rapides d'exécution. Elles sont moins précises que les méthodes basées sur les fonctions spline et leurs dérivées ne sont pas necéssairement continues. Une méthode utilisant le filtre de Kalman afin de faire l'approximation d'une fonction spline cubique pourrait fondre ensemble les meilleurs éléments de chaque méthode.

III. Les fonctions Spline

Lorsque nous parlons d'échantillons ou de points, nous avons souvent à travailler avec des méthodes d'interpolation. L'interpolation permet d'évaluer l'état d'un phénomène entre deux points d'échantillonnage. Plusieurs méthodes existent pour effectuer cette opération. La plus simple est l'interpolation linéaire. Quoique simple, rares sont les phénomènes physiques qui se comportent comme le prédit l'interpolation linéaire. En effet, pratiquement tout phénomène empêche des changements d'état brutaux. Par le fait même, la première et souvent la deuxième dérivée d'une variable est continue, et non discontinue comme le suggère une interpolation linéaire. Pour un signal composé de N points, nous pouvons toujours définir une fonction d'ordre N pour faire l'interpolation. Cette méthode est lourde et crée beaucoup d'oscillations dans les extrémités du signal. Une autre alternative est la fonction spline. Ce chapitre donne une partie de la théorie nécessaire pour apprécier ce qui suivra.

Les fonctions spline ont longtemps été utilisées dans des domaines tels que le graphisme et le dessin assisté par ordinateur. L'origine du terme vient, selon le dictionnaire Oxford, d'un outil utilisé par les dessinateurs consistant en une bande de caoutchouc flexible appelé spline, utilisée pour tracer des courbes entre une série de points. Ce même dictionnaire définit la fonction spline comme étant une courbe continue construite de façon à passer par une série de points tout en ayant sa première et deuxième dérivées continues. La figure 5 montre un exemple possible.

Fig 5: Fonction spline unissant 4 points.

Mathématiquement, ces fonctions peuvent être définies de plusieurs façons. On propose ici la méthode de représentation la plus simple pour bien comprendre son fonctionnement. Plus loin, nous devrons décrire une nouvelle notification afin d'utiliser moins de mémoire et du même coup permettre l'établissement d'une représentation d'état du système de mesure.

Si x = f (t) est une fonction spline cubique, on peut écrire:

$$x(t) = \sum_{n=0}^{N-1}\left[\mathbf{1}(t-t_n) - \mathbf{1}(t-t_{n+1})\right]\left\{A_n(t-t_n)^3 + B_n(t-t_n)^2 + C_n(t-t_n) + D_n\right\} \qquad (8)$$

où $\mathbf{1}(\tau)$ est un échelon unitaire.

Alors N+1 est le nombre de points dans la série définissant l'allure générale de la courbe. La fonction est divisée en N segments bornés par deux points consécutifs de la série. Évidemment, t_n est la valeur du temps où s'est produit le $n^{ième}$ point. Dans le domaine de la métrologie (et dans plusieurs autres domaines) un point est souvent appelé échantillon et vient d'une discrétisation inévitable lors d'un mesurage. Aussi, le temps entre deux échantillons est habituellement constant; on l'appelle temps d'échantillonnage et on le note Δt. Finalement, les coefficients A_n, B_n, C_n et D_n sont des paramètres décrivant l'allure du segment existant entre le point n et n+1.

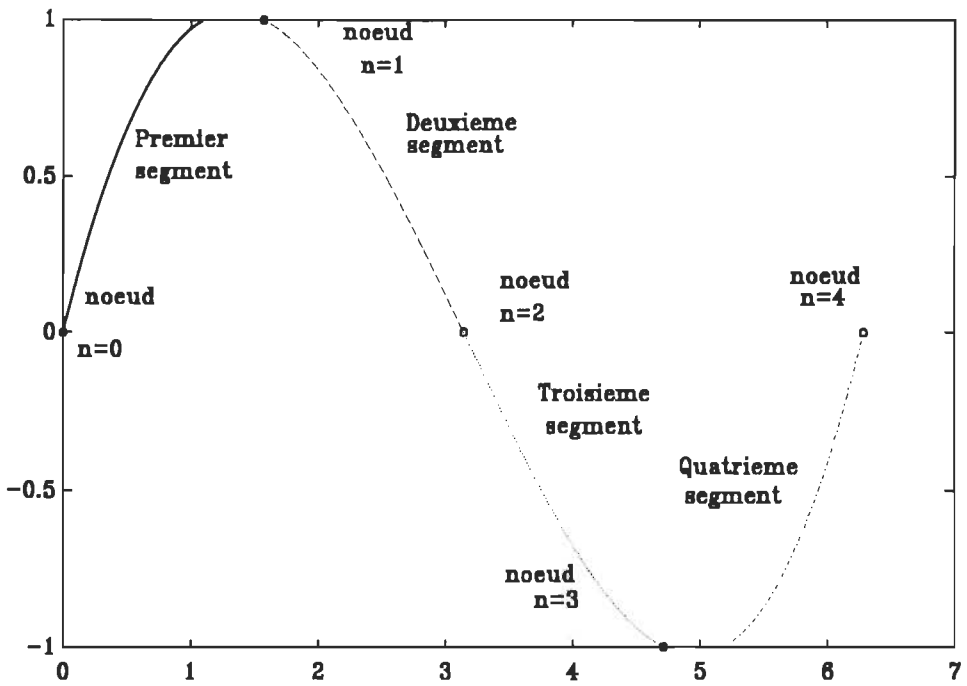


Fig 6: Quatre segments d'une fonction spline cubique.

La figure 6 montre une fonction spline utilisant quatre segments (N=4) approximant une sinusoïde. Les 5 noeuds (N+1 points) sont 0, $\pi/2$, π, $3\pi/2$, 2π. A remarquer qu'il n'y a pas de réelle saturation au maxima et minima mais que le logiciel

utilisé crée du mieux qu'il peut les axes et y confine les données. Les coefficients pour chaque segment sont présentés à la table I.

Table I: Coefficients spline des segments.

Segment	n	A	B	C	D
0 - π/2	0	0.0860	-0.8106	1.6977	0
π/2 - π	1	0.0860	-0.4053	-0.2122	1
π - 3π/2	2	0.0860	0	-0.8488	0
3π/2 - 2π	3	0.0860	0.4053	-0.2122	-1

On comprend déjà pourquoi une méthode de reconstitution utilisant les fontions spline peut demander beaucoup de mémoire lors d'un traitement numérique. En effet, à chaque échantillon sont associés quatre paramètres afin de définir son segment. De plus, certaine méthodes demandent des calculs matriciels où la matrice de base est carrée[2]. Si par exemple on veut traiter un signal ayant 256 échantillons (c'est une quantité standard), il faudra $(4 \text{ X } 256)^2 = 1\ 048\ 576$ éléments pour contenir la matrice. Sachant que chaque élément est un nombre en point flottant nécessitant 2, 4, et parfois même 8 octets pour le contenir (selon le système: Dos, Unix, DSP, etc...), on s'aperçoit vite de l'impressionnante quantité de mémoire necéssaire pour cette seule matrice, sans parler du temps de traitement!

Mais comme on l'a déjà mentionné, la précision et la continuité des courbes résultantes de ces méthodes sont très intéressantes pour certaines applications. Pour obtenir cette continuité, les paramètres doivent être judicieusement choisis. L'application des contraintes suivantes permet de bien évaluer les paramètres.

$$D_{n+1} = D_n + C_n\Delta t + B_n\Delta t^2 + A_n\Delta t^3 \quad (9)$$

$$C_{n+1} = C_n + 2B_n\Delta t + 3A_n\Delta t^2 \quad (10)$$

$$B_{n+1} = B_n + 3A_n\Delta t \quad (11)$$

Par exemple, utilisons les valeurs de la table I au point π. Alors: $\Delta T = \pi/2$

$$D_2 = 0 = 1 - 0.2122 * \pi/2 - 0.4053 * (\pi/2)^2 + 0.0860 * (\pi/2)^3$$

$$0 = 1 - 0.333 - 1 + 0.333$$

$C_2 = -0.8488 = -0.2122 - 2 * 0.4053 * \pi/2 + 3 * 0.086 * (\pi/2)^2$

$B_2 = 0 = -0.4053 + 3 * 0.086 * \pi/2$

Les paramètres du deuxième segment sont donc dépendants du premier segment, ceux du troisième dépendent du deuxième et ainsi de suite. Plusieurs algorithmes existent déja afin d'évaluer ces paramètres; un "toolbox" existe d'ailleurs pour Matlab. Alors pourquoi se casser la tête? Rappelons notre problème: nous voulons définir les paramètres d'une fonction spline représentant l'allure d'un mesurande que l'on ne peut pas échantillonner directement car notre instrument de mesure l'a déformé et bruité.

Nous avons vu une façon de définir une fonction spline cubique. C'est la plus commune. Nous pouvons, à cause du lien entre les segments, définir la fonction selon les valeurs de celle-ci au noeud (x_n) et selon ses dérivés ($\dot{x}_n$). Pour ce faire, nous pouvons écrire selon (9) et (10):

$$D_n = x_n \tag{12}$$

$$C_n = \dot{x}_n \tag{13}$$

$$A_n \Delta t^3 + B_n \Delta t^2 + C_n \Delta t + D_n = x_{n+1} \tag{14}$$

$$3A_n \Delta t^2 + 2B_n \Delta t + C_n = \dot{x}_{n+1} \tag{15}$$

En isolant B_n de (14) et (15) afin que les équations soient égales, tout en tenant compte de (12) et (13), nous trouvons:

$$A_n = \frac{1}{\Delta t^2}\left(\dot{x}_{n+1} + \dot{x}_n - 2\frac{x_{n+1} - x_n}{\Delta t} \right) \tag{16}$$

En isolant A_n de (14) et (15) afin que les équations soient égales, tout en tenant compte de (12) et (13), nous trouvons:

$$B_n = \frac{1}{\Delta t}\left(3\frac{x_{n+1} - x_n}{\Delta t} - \dot{x}_{n+1} - 2\dot{x}_n \right) \tag{17}$$

et bien sur, de (12) et (13), on a simplement:

$$C_n = \dot{x}_n \tag{18}$$

et

$$D_n = x_n \tag{19}$$

Après une simple manipulation algébrique soit la substitution de (16), (17), (18) et (19) dans (8), alors cette dernière pourra être transformée comme suit:

$$\hat{x}(t) = \sum_{n=0}^{N-1}\left[\mathbf{1}(t-t_n) - \mathbf{1}(t-t_{n+1})\right]\cdot \left\{W_{00}(t-t_n)x_n + W_{01}(t-t_n)\dot{x}_n + W_{10}(t-t_n)x_{n+1} + W_{11}(t-t_n)\dot{x}_{n+1}\right\} \tag{20}$$

où $$W_{00}(u) = 2\frac{u^3}{\Delta t^3} - 3\frac{u^2}{\Delta t^2} + 1 \tag{21}$$

$$W_{01}(u) = \frac{u^3}{\Delta t^2} - 2\frac{u^2}{\Delta t} + u \tag{22}$$

$$W_{10}(u) = -2\frac{u^3}{\Delta t^3} + 3\frac{u^2}{\Delta t^2} \tag{23}$$

$$W_{11}(u) = \frac{u^3}{\Delta t^2} - \frac{u^2}{\Delta t} \tag{24}$$

Cette transformation nous sera très utile car en plus de diminuer l'espace mémoire requis pour définir la fonction spline, x_n et $\dot{x}_n$ plutôt que An, Bn, Cn et Dn, elle intègre les contraintes directement à la fonction.

IV. Le filtre de Kalman

Le filtre de Kalman est né d'un mariage entre l'analyse par modèle d'état et les notions de statistique. Son intérêt et sa popularité viennent du fait qu'il fait un traitement en temps réel (selon les capacités du calculateur). Chaque échantillon obtenu est donc traité avant même d'obtenir le prochain échantillon. Par ce filtre, tout système pouvant être exprimé sous forme d'équations d'état probabiliste, tel que:

$$z_{k+1} = \mathbf{F}_k z_k + \mathbf{b}_k u_k \tag{25}$$

$$y_k = \mathbf{h}_k^T z_k + \eta_k \tag{26}$$

peut être traité suivant des paramètres statistiques connus du système. Ces paramètres doivent répondre aux conditions suivantes:

- u_k sont des réalisations d'une variable aléatoire $\underline{u}_k$ identique et de moyenne nulle et de plus une estimation $\breve{\sigma}_u^2$ de leur variance σ_u^2 est disponible a-priori;
- η_k sont des réalisations d'une variable aléatoire $\underline{\eta}_k$ identique et de moyenne nulle et de plus une estimation $\breve{\sigma}_\eta^2$ de leur variance σ_η^2 est disponible a-priori;
- les variables $\underline{u}_k$ et $\underline{\eta}_k$ sont non-correlées, i.e.: $E\left[\underline{u}_n \underline{\eta}_m\right] = 0$, $E\left[\underline{u}_n \underline{u}_m\right] = \sigma_u^2 \delta_{nm}$ et $E\left[\underline{\eta}_n \underline{\eta}_m\right] = \sigma_\eta^2 \delta_{nm}$ pour tout n et m $\in$ IN*

L'explication complète du fonctionnement de ce filtre est plutôt ardue et déborde le champ de ce travail. Le lecteur intéressé pourra consulter Anderson-Moore [3]. Disons seulement que plusieurs méthodes existent pour implanter ce filtre. Les équations de base sont:

$$\Sigma_{k+1/k} = \mathbf{F}_k \Sigma_{k/k} \mathbf{F}_k^T + \mathbf{b}_k q_k \mathbf{b}_k^T \tag{27}$$

$$r_{k+1}^e = r_{k+1} + \mathbf{h}_{k+1}^T \Sigma_{k+1/k} \mathbf{h}_{k+1} \tag{28}$$

$$\mathbf{k}_{k+1} = \Sigma_{k+1/k} \mathbf{h}_{k+1} (r_{k+1}^e)^{-1} \tag{29}$$

$$\Sigma_{k+1/k+1} = (\mathbf{I} - \mathbf{k}_{k+1} \mathbf{h}_{k+1}^T) \Sigma_{k+1/k} \tag{30}$$

$$z_{k+1/k+1} = \mathbf{F}_k z_{k/k} + \mathbf{k}_{k+1} (y_{k+1} - \mathbf{h}_{k+1}^T \mathbf{F}_k z_{k/k}) \tag{31}$$

où **F**, **b**, **h**, z, et y sont les éléments des équations d'état ci-haut;

q et r sont des variances de bruit (σ_u^2 et σ_η^2);

Σ est la matrice de covariance;

I est la matrice identité;

k est le vecteur de gain de Kalman;

k est le numéro de l'échantillon en traitement;

$\mathbf{h}^T$ représente la transposée de **h**;

$z_{k+1/k}$ donne z au moment k+1 connaissant z aux moments k.

Intuitivement, on comprend de (31) que le nouvel échantillon ($z_{k+1/k+1}$) est calculé à partir du dernier échantillon ($z_{k/k}$) additionné à une erreur ($y_{k+1} - \mathbf{h}_{k+1}^T \mathbf{F}_k z_{k/k}$), appelé innovation, multiplié par un certain gain ($\mathbf{k}_{k+1}$). Les équations (27), (28), (29) et (30) servent au calcul du gain pour cet échantillon ainsi qu'au calcul de la matrice de covariance pour le prochain échantillon. Le vecteur de gain de Kalman fait en sorte de minimiser l'espérance de manière statistique:

$$E\left[(z_{k+1} - z_k)(z_{k+1} - z_k)^T\right] \tag{32}$$

Cette forme du filtre de Kalman est très souple. En effet, il peut traiter des systèmes variants et non-stationnaires; i.e. **F** et/ou **h** peuvent changer avec le temps, mais puisque ces valeurs font partie du calcul du gain, alors ce dernier s'ajuste dynamiquement avec peu d'erreur. Lorsqu'un système est invariant et stationnaire on peut modifier l'algorithme afin de le rendre plus rapide et plus précis. Pour faire cela, on peut précalculer un vecteur de gain **k** pour $k = \infty$, c'est à dire après plusieurs échantillons. On applique ensuite toujours ce même gain dans (31) pour tous les échantillons.

V. L'algorithme proposé

Généralités :

Comme nous l'avons déja souligné, la théorie nous dit qu'un signal (x) entrant dans un système ayant une réponse impulsionnelle (g) donne un signal de sortie (y) qui est la convolution de x et g [4]. Le bruit (η) est alors ajouté pour donner un signal de sortie perturbé ($\tilde{y}$). Le modèle échantillonné de cette convolution est donné par:

$$\tilde{y}_n = \int_0^{t_n} g(t_n - \tau)\cdot x(\tau)\,d\tau + \eta_n \qquad \text{où } t_n = n\Delta t \qquad n = 0, 1, \ldots, N \tag{33}$$

L'idée de l'algorithme étudié dans ce mémoire est de représenter le signal d'entrée par une fonction spline, alors de (8) on obtient:

$$\hat{x}(t) = \sum_{n=0}^{N-1}\left[\mathbf{1}(t - t_n) - \mathbf{1}(t - t_{n+1})\right]\left\{A_n(t - t_n)^3 + B_n(t - t_n)^2 + C_n(t - t_n) + D_n\right\} \tag{34}$$

Les contraintes peuvent être intégrées à cette fonction grâce au développement dont nous avons déja parlé; soit les équations (16), (17), (18) et (19). Ce changement permet de représenter la fonction spline par les valeurs d'échantillon (x_n où: $n \in [0, N]$) ainsi que par les dérivées de la fonction aux temps d'échantillonnage ($\dot{x}_n$ où: $n \in [0, N]$).

En conséquence:

$$\hat{x}(t) = \sum_{n=0}^{N-1}\left[\mathbf{1}(t - t_n) - \mathbf{1}(t - t_{n+1})\right]\cdot \left\{W_{00}(t - t_n)x_n + W_{01}(t - t_n)\dot{x}_n + W_{10}(t - t_n)x_{n+1} + W_{11}(t - t_n)\dot{x}_{n+1}\right\} \tag{35}$$

où: $$W_{00}(u) = 2\frac{u^3}{\Delta t^3} - 3\frac{u^2}{\Delta t^2} + 1 \tag{36}$$

$$W_{01}(u) = \frac{u^3}{\Delta t^2} - 2\frac{u^2}{\Delta t} + u \tag{37}$$

$$W_{10}(u) = -2\frac{u^3}{\Delta t^3} + 3\frac{u^2}{\Delta t^2} \tag{38}$$

$$W_{11}(u) = \frac{u^3}{\Delta t^2} - \frac{u^2}{\Delta t} \tag{39}$$

Maintenant, si nous reprenons notre convolution avant l'ajout du bruit et en utilisant notre modèle basé sur spline, nous obtenons:

$$y_n = \sum_{\upsilon=0}^{n-1} \int_{t_\upsilon}^{t_{\upsilon+1}} g(t_n - \tau)\left[W_{00}(\tau - t_\upsilon)x_\upsilon + W_{01}(\tau - t_\upsilon)\dot{x}_\upsilon + W_{10}(\tau - t_\upsilon)x_{\upsilon+1} + W_{11}(\tau - t_\upsilon)\dot{x}_{\upsilon+1}\right]d\tau \tag{40}$$

ou plus simplement:

$$y = \sum_{\upsilon=0}^{n-1}\left(h^{00}_{n-\upsilon}x_\upsilon + h^{01}_{n-\upsilon}\dot{x}_\upsilon + h^{10}_{n-\upsilon}x_{\upsilon+1} + h^{11}_{n-\upsilon}\dot{x}_{\upsilon+1}\right) \tag{41}$$

avec $$h^{ij}_k = \int_0^{\Delta t} g(t_k - u)W_{ij}(u)du \tag{42}$$

Le vecteur **h** est donc la convolution de la réponse impulsionnelle (g) avec chaque contrainte (W) introduite par l'utilisation de la fonction spline. Afin de rendre les équations (41) et (42) plus concises nous pouvons écrire:

$$y = h^{10}_1 x_n + h^{11}_1 \dot{x}_n + \sum_{\upsilon=1}^{n-1}\left(h^0_\upsilon x_{n-\upsilon} + h^1_\upsilon \dot{x}_{n-\upsilon}\right) + h^{00}_n x_0 + h^{01}_n \dot{x}_0 \tag{43}$$

où $$h^0_\upsilon = h^{00}_\upsilon + h^{10}_{\upsilon+1} \tag{44}$$

$$h^1_\upsilon = h^{01}_\upsilon + h^{11}_{\upsilon+1} \tag{45}$$

Nous pouvons écrire matriciellement:

$$y_n = \mathbf{h}_n^T \mathbf{v}_n \tag{46}$$

où $$\mathbf{h}_n^T = \left[h^{10}_1 \middle| h^{11}_1 \middle| h^0_1 \middle| h^1_1 \middle| \cdots \middle| h^0_{n-1} \middle| h^1_{n-1} \middle| h^0_n \middle| h^1_n \middle| h^{00}_n \middle| h^{01}_n \middle| 0 \middle| \cdots \middle| 0\right] \quad \dim(\mathbf{h}) = 2N$$

$$\mathbf{v}_n = \left[x_n \middle| \dot{x}_n \middle| x_{n-1} \middle| \dot{x}_{n-1} \middle| \cdots \middle| x_1 \middle| \dot{x}_1 \middle| x_0 \middle| \dot{x}_0 \middle| 0 \middle| \cdots \middle| 0\right] \quad \dim(\mathbf{v}) = 2N$$

Pour n = 1, 2, ..., N

Le nombre de zéro remplissant la fin des vecteurs $\mathbf{h}_n$ et $\mathbf{v}_n$ dépendent de la valeur de n. Aussi, lorsque n=N, il n'y a plus de zéro dans $\mathbf{h}_n$ et les éléments x0 et $\dot{x}_0$ sont expulsés du vecteur $\mathbf{v}_n$ de même que les zéros.

L'équation (46) est en fait la convolution de la réponse impulsionnelle (g), qui est contenue dans le vecteur **h** selon (42), et du signal d'entrée (x). Par l'équation (42) on a pu isoler les coefficients de la fonction spline ($\mathbf{v}_n$) de l'intégrale de convolution. Ainsi, connaissant g nous pouvons précalculer toutes les valeurs du vecteur **h**.

Modèle de deuxième ordre :

Maintenant, pour établir une équation d'état pouvant être traitée par le filtre de Kalman et modélisant notre canal de mesure, on peut supposer que la deuxième dérivée de $\hat{x}$ est une réalisation d'un processus aléatoire dont la moyenne est nulle (c'est une condition d'application du filtre de Kalman p.15). Nous obtenons ainsi un filtre de deuxième ordre. Alors:

$$\frac{\dot{x}_{n+1} - \dot{x}_n}{\Delta t} = \xi_n \quad \text{ou} \quad \dot{x}_{n+1} = \dot{x}_n + \Delta t \xi_n \tag{47}$$

où ξ_n est une réalisation d'une variable aléatoire, telle que sa moyenne égale zéro selon les conditions déjà énumérées au Chap. IV. La génération de la séquence $\{x_n\}$ est décrite par l'équation:

$$x_{n+1} = x_n + \dot{x}_n \Delta t + \frac{1}{2}\ddot{x}_n \Delta t^2 \cong x_n + \dot{x}_n \Delta t + \frac{1}{2}\xi_n \Delta t^2 \tag{48}$$

En conséquence, le modèle d'état complet prend la forme matricielle:

$$\mathbf{v}_{n+1} = \Phi \mathbf{v}_n + \mathbf{b}\xi_n \tag{49}$$

$$y_n = \mathbf{h}_n^T \mathbf{v}_n + \eta_n \tag{50}$$

où η_n est une réalisation de bruit s'ajoutant à la sortie et répondant aussi aux conditions énumérées au Chap. IV

et $$\Phi = \begin{bmatrix} 1 & \Delta t & 0 & & \cdots & & 0 \\ 0 & 1 & 0 & & & & \\ 1 & 0 & 0 & & \ddots & & \vdots \\ 0 & 1 & 0 & & & & \\ 0 & 0 & 1 & & & & \\ \vdots & \ddots & & \ddots & 0 & 0 & 0 \\ 0 & 0 & \cdots & 0 & 1 & 0 & 0 \end{bmatrix} \qquad \mathbf{b} = \begin{bmatrix} \Delta t^2 / 2 \\ \Delta t \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Les deux premiers rangs de la matrice Φ sont en fait l'application des équations (48) et (47) respectivement. Le reste des rangs ne sert qu'à causer un décalage afin que x_n devienne x_{n-1}, que x_{n-1} devienne x_{n-2},etc...

Modèle de premier ordre :

Nous pouvons aussi modéliser notre canal de mesure en supposant que la première dérivée de $\hat{x}$ est une réalisation d'un processus aléatoire dont la moyenne est nulle et obtenir un filtre d'ordre un. Alors:

$$\dot{x}_{n+1} = \xi_n \tag{51}$$

où ξ_n est une réalisation d'une variable aléatoire, telle que sa moyenne égale zéro. La génération de la séquence $\{x_n\}$ est décrite par l'équation:

$$x_{n+1} = x_n + \dot{x}_n \Delta t + \frac{1}{2}\ddot{x}_n \Delta t^2 \cong x_n + \dot{x}_n \Delta t + \frac{1}{2}\xi_n \tag{52}$$

Dans ce cas , on obtient:

$$\Phi = \begin{bmatrix} 1 & \Delta t & 0 & & \cdots & & 0 \\ 0 & 0 & 0 & & & & \\ 1 & 0 & 0 & & \ddots & & \vdots \\ 0 & 1 & 0 & & & & \\ 0 & 0 & 1 & & & & \\ \vdots & \ddots & & \ddots & 0 & 0 & 0 \\ 0 & 0 & \cdots & 0 & 1 & 0 & 0 \end{bmatrix} \qquad \mathbf{b} = \begin{bmatrix} 1 / 2 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Modèle de continuité :

Finalement, un dernier modèle a été étudié. Il permet de tenir compte de la continuité de la deuxième dérivée que permet la fonction spline. Pour ce faire, les équations (16) et (17), liant les coefficients A et B aux valeurs des noeuds et leurs dérivés, sont intégrées à l'équation forçant la continuité de la deuxième dérivée; soit l'équation (11). Après développement, on peut isoler x_{n+1}:

$$x_{n+1} = x_{n-1} + \frac{1}{3}\dot{x}_{n-1}\Delta t + \frac{4}{3}\dot{x}_n\Delta t + \frac{1}{3}\dot{x}_{n+1}\Delta t \tag{53}$$

où $\dot{x}_{n+1}\Delta t = \xi_n$ (filtre de premier ordre) (54)

et ξ_n est une réalisation d'une variable aléatoire, telle que sa moyenne égale zéro. Dans ce cas , on obtient:

$$\Phi = \begin{bmatrix} 0 & 4/3\Delta t & 1 & 1/3\Delta t & 0 & \cdots & 0 \\ 0 & 0 & 0 & 0 & 0 & \cdots & 0 \\ 1 & 0 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & & \ddots & & \ddots & 0 \\ 0 & 0 & \cdots & 0 & 1 & 0 & 0 \end{bmatrix} \qquad \mathbf{b} = \begin{bmatrix} 1/3\Delta t \\ 1 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Application du filtre de Kalman au modèle :

Trois modèles existent donc pour notre canal de mesure. D'autres auraient pu être créés mais ces trois-ci sont très simples et semblent les plus prometteurs.

Nous pouvons maintenant appliquer le filtre de Kalman au modèle d'état afin d'obtenir le vecteur $\hat{\mathbf{v}}_n$ qui définit tous les coefficients de la fonction spline définissant $\hat{\mathbf{x}}$. Comme on a vu au chapitre précédent, plusieurs méthodes existent pour appliquer le filtre de Kalman. On peut appliquer l'algorithme dynamique mais, dans notre cas, on peut aussi appliquer le filtre statique puisque Φ et g sont constants dans le temps.

Voyons premièrement l'allure du filtre dynamique. On obtient dans ce cas de (27), (28), (29), (30) et (31), les équations simplifiées suivantes:

$$\Sigma_{n+1-} = \Phi\Sigma_n\Phi^T + \sigma_\xi^2 \mathbf{b}\mathbf{b}^T \tag{55}$$

$$\Sigma_{n+1} = \Sigma_{n+1-} - \frac{1}{\sigma_\eta^2 + \mathbf{h}_n^T\Sigma_{n+1-}\mathbf{h}_n}\Sigma_{n+1-}\mathbf{h}_n\mathbf{h}_n^T\Sigma_{n+1-} \tag{56}$$

$$\mathbf{k}_{n+1} = \frac{1}{\sigma_\eta^2}\Sigma_{n+1}\mathbf{h}_n \tag{57}$$

$$\hat{\mathbf{v}}_{n+1} = \Phi\hat{\mathbf{v}}_n + \mathbf{k}_{n+1}(\tilde{y}_n - \mathbf{h}_n^T\Phi\hat{\mathbf{v}}_n) \tag{58}$$

où σ_ξ^2 est la variance du générateur de bruit décrivant le signal d'entrée;

σ_η^2 est la variance du bruit ajoutée au signal de sortie.

Ces deux variances sont souvent regroupées en un seul paramètre, soit:

$$\beta = \frac{\sigma_\xi^2}{\sigma_\eta^2} \tag{59}$$

On peut ainsi réécrire les équations (42) à (44) comme suit:

$$\Sigma_{n+1-} = \Phi\Sigma_n\Phi^T + \beta\mathbf{b}\mathbf{b}^T \tag{60}$$

$$\Sigma_{n+1} = \Sigma_{n+1-} - \frac{1}{1 + \mathbf{h}_n^T\Sigma_{n+1-}\mathbf{h}_n}\Sigma_{n+1-}\mathbf{h}_n\mathbf{h}_n^T\Sigma_{n+1-} \tag{61}$$

$$\mathbf{k}_{n+1} = \Sigma_{n+1}\mathbf{h}_n \tag{62}$$

Ainsi, pour faire la reconstitution d'un signal modélisé par une fonction spline en utilisant un filtre de Kalman, chaque échantillon du signal brut de mesure ($\tilde{y}$) est appliqué aux équations ci-haut. À chaque application, le vecteur de gain évolue vers un vecteur de gain optimal. Aussi, le vecteur **h** s'allonge commençant avec deux termes et s'arrêtant à 2N termes. Donc, lors de la première application des équations il faut que:

$$\mathbf{h}_0^T = \left[h_0^{00} \middle| h_0^{01} \middle| 0 \middle| \cdots \middle| 0 \right] \qquad \text{calculé selon l'équation (42)}$$

$$\hat{\mathbf{v}}_0^T = \left[\hat{x}_0 \middle| \hat{\dot{x}}_0 \middle| 0 \middle| \cdots \middle| 0 \right] \qquad \text{estimées de x(0) et } \dot{x}(0)$$

et $$\Sigma_0 = \begin{bmatrix} \sigma_0^2 & 0 & 0 & \cdots & 0 \\ 0 & \sigma_1^2 & 0 & & \\ 0 & 0 & 0 & & \vdots \\ \vdots & & & \ddots & \\ 0 & & \cdots & & 0 \end{bmatrix}$$ variances des erreurs des estimations

Application du filtre de Kalman stationnaire :

Si on veut simplifer le calcul, on peut prendre le traitement stationnaire. Il faut rappeler qu'il s'agit alors de précalculer un vecteur de gain $\mathbf{k}_n$ pour $n=\infty$, i.e. après plusieurs échantillons. On applique ensuite toujours ce même gain avec $\mathbf{h} = \mathbf{h}_N$ pour tous les échantillons. Donc, si:

$$\Sigma_\infty = \lim_{i \to \infty} \Sigma_i \tag{63}$$

partant de Σ_0 vu précédement,

$$\Sigma_{i+1} = \left[\left(\Phi \Sigma_i \Phi^T + \sigma_\xi^2 \mathbf{b}\mathbf{b}^T \right)^{-1} + \frac{1}{\sigma_\eta^2} \mathbf{h}\mathbf{h}^T \right]^{-1} \tag{64}$$

lorsqu'il y a convergence de Σ alors on calcule le gain:

$$\mathbf{k}_\infty = \frac{1}{\sigma_\eta^2} \Sigma_\infty \mathbf{h} \tag{65}$$

Pour chaque échantillon, on peut simplement appliquer l'équation (58) en utilisant toujours $\mathbf{k}_\infty$:

$$\hat{v}_{n+1} = \Phi \hat{v}_n + \mathbf{k}_\infty (\tilde{y}_n - \mathbf{h}^T \Phi \hat{v}_n) \tag{66}$$

Nous découvrons à l'usage que l'équation (64) peut causer des difficultés. En effet, selon la forme de $\mathbf{\Phi}$ et **b,** l'inversion de matrice contenue dans l'équation (64) peut causer une singularité. Pour contourner le problème, on peut calculer $\mathbf{k}_\infty$ en réappliquant (55), (56) et (57) itérativement jusqu'à convergence de **k**.

Application d'une contrainte de positivité :

Finalement, comme dans d'autres méthodes de reconstitution de mesurande, il est parfois utile d'instaurer une contrainte particulière au traitement afin d'améliorer le résultat de reconstitution. Dans le cas de signaux spectrométriques entre autres, il est possible d'ajouter une contrainte de positivité[5]. Cette dernière force le résultat de reconstitution à avoir des valeurs négatives moins grandes. Dans le cas de notre algorithme, la contrainte de positivité peut être appliquée en ajoutant dans la boucle de traitement (ceci défini la valeur de n) la condition suivante:

$$\hat{v}_{n,2v+1} = \begin{cases} c\hat{v}_{n,2v+1} & \text{si} \quad \hat{v}_{n,2v+1} \le 0 \\ \hat{v}_{n,2v+1} & \text{si} \quad \hat{v}_{n,2v+1} > 0 \end{cases} \qquad \text{pour } v = 0, 1, \ldots, N \qquad (67)$$

avec $0 \le c \le 1$.

ou: $\hat{v}_{n,2v+1}$ est le $2v+1$ iem élément du vecteur $\hat{v}_n$.

Ainsi, seuls les échantillons de reconstitution de valeur négative seront ajustés vers une valeur plus près de zéro, donc moins négatif, et leurs dérivées ne seront pas altérées. On devine déjà que pour certains signaux cette contrainte pourrait être très nuisible puisque spline demande la continuité des dérivées. Pour cela, après l'application de la contrainte, il est possible de réajuster les dérivées soit par:

$$\dot{x}_n = \frac{x_n - x_{n-1}}{\Delta t} \qquad (68)$$

ou par:

$$\dot{x}_n = \frac{x_{n+1} - x_{n-1}}{2\Delta t} \qquad (69)$$

Il est important de noter ici qu'il ne faut pas réajuster la dérivée du dernier échantillon traité car ceci arrêterait le bon fonctionnement du filtre de Kalman.

Ordre de grandeur des variances demandées :

Parlons enfin des valeurs à donner aux variances. Ces valeurs doivent être estimées. L'avantage de l'utilisation du paramètre β plutôt que des deux variances, est bien sûr de diminuer le nombre de paramètres à estimer. Mais pour estimer correctement le paramètre β, il est bon de comprendre la source des deux variances. Pour ce qui est de la variance du bruit (σ_η^2) ajouté au signal de sortie, elle dépend de l'instrument de mesure ainsi que de la qualité électromagnétique de l'environnement. Un laboratoire de mesure peut avoir une variance de l'ordre de 1e-6 tandis qu'une mesure faite aux environs d'un moteur à courant continu de qualité moyenne alimenté par des MOSFETs pourrait provoquer un bruit ayant une variance de l'ordre de 1e-1.

La variance du générateur de bruit (σ_ξ^2) décrivant le signal d'entrée est moins simple à évaluer. Lors de simulations, puisque les valeurs de x sont connues à l'avance, on peut directement calculer cette variance par:

$$\sigma_\xi^2 = \frac{\sum_{n=0}^{N-1}\left(\frac{\dot{x}_{n+1} - \dot{x}_n}{\Delta t}\right)^2}{N} \qquad \text{pour un filtre d'ordre deux} \qquad (70)$$

et

$$\sigma_\xi^2 = \frac{\sum_{n=0}^{N-1} \dot{x}_n^{\,2}}{N} \qquad \text{pour un filtre d'ordre un.} \qquad (71)$$

Malheureusement, en pratique on ne peut pas utiliser ces formules car le signal x n'est pas connu, on le cherche. On peut cependant appliquer (70) au signal de sortie ($\tilde{y}$) et faire une première reconstitution avec la variance obtenue. On réapplique (70) avec le résultat de reconstitution puis on reprend une deuxième reconstitution avec cette nouvelle valeur de variance. On peut continuer comme cela tant qu'il n'y a pas convergence du signal reconstitué ($\hat{x}$).

VI. Développement de l'algorithme

L'algorithme est d'abord implanté sur Matlab. Le traitement de matrice de façon implicite qu'offre ce logiciel, en fait l'outil idéal pour le développement d'un tel algorithme. Le premier essai d'implantation a été de programmer intégralement les formules du filtre non-stationnaire de Kalman. On crée avant tout un vecteur d'entrée (x) et un transopérateur (g). On fait une convolution discrète des deux signaux pour obtenir le signal de sortie pur (y). Ce dernier est ensuite bruité avec une série de nombres aléatoires d'une certaine amplitude. Cette amplitude donne une idée de la variance du bruit. On a maintenant un signal de sortie brut ($\tilde{y}$). On crée aussi les matrices de départ contenant surtout des zéros. L'algorithme traite donc ces matrices et vecteurs. Ceux-ci évoluent à chaque pas de calcul; les zéros sont remplacés par des estimations et il y a bien sûr autant de pas que d'échantillons à traiter. Le résultat de reconstitution ($\hat{x}$) est mis sous forme de graphique et comparé au signal d'entrée original (x). L'erreur relative est calculée selon:

$$\varepsilon = \sqrt{\frac{\sum(\hat{x}-x)^2}{\sum x^2}} \tag{72}$$

Que peut-on faire de plus? Et bien, l'algorithme implanté est très lent et beaucoup de ce temps est perdu, lors du traitement des premiers échantillons, à calculer des zéros! L'algorithme est alors modifié pour calculer au départ avec de simples matrices de 2X2 et des vecteurs de 2 éléments. Au deuxième pas, les matrices ont 4X4 et les vecteurs 4 éléments et ainsi de suite jusqu'à une longueur de 2N. La matrice Φ peut aussi disparaître car on peut faire les translations autrement et on applique directement les équations décrivant le modèle du canal de mesure. Cette dernière remarque, sans épargner de temps, permet tout de même de sauver beaucoup de mémoire.

Ces premiers essais d'implantation donnent des résultats acceptables et nous encouragent à tenter d'améliorer la méthode. Comme on l'a déja mentionné, pour beaucoup d'applications, le filtre stationnaire est plus avantageux. C'est donc l'étape suivante.

Un premier processus calcule les vecteurs $\mathbf{k}_\infty$ et $\mathbf{h}$, un deuxième utilise ces vecteurs et applique l'equation (58) pour chaque point. De plus, on peut, après le calcul du gain de Kalman $\mathbf{k}_\infty$ et $\mathbf{h}$, sauvegarder ces derniers sur disque. Ainsi, si une

reconstitution semblable (même fonction de transfert et même paramètre β) doit être faite eplus tard, on sera dispensé de recalculer ces vecteurs.

La dernière étape a été d'intégrer l'algorithme stationnaire fait sur Matlab au système SÉR [6] (Simulation, Étallonage, Reconstitution) du laboratoire de système de mesure. L'avantage découlant de cette agglutination est la facilité de génération de signaux à traiter. Il faut tout de même bien connaître les signaux générés car certaines de leurs caractéristiques ne sont pas tel qu'on pourait le penser et donc peuvent causer beaucoup de problèmes! Une vaste librairie de signaux est intégrée à SCR. L'ajout de différents niveaux de bruit est un jeu d'enfant et l'affichage automatisé des résultats est très plaisant: graphiques sur quatre fenêtres, comparaison de courbes, calcul d'erreur relative, ajout de texte sur les graphiques, etc... Cette intégration à SCR est très simple; il s'agit d'ajouter l'appel à notre routine et, bien sûr, d'inclure cette routine au programme. Il faut aussi modifier les menus affichés par SCR afin de donner à l'usager la possibilité d'appeler la nouvelle méthode de reconstitution. Ce dernier changement est fait interactivement à partir du système SCR.

Donc, en résumé, les algorithmes stationnaires d'ordre 1 et d'ordre 2 sont implantés dans le système SCR autant en langage C qu'en Matlab et permettent l'incorporation d'une contrainte de positivité de force réglable (paramètre c de l'equation (67)). L'algorithme a été développé en Matlab au départ, afin de trouver une façon demandant le moins de mémoire possible, tout en étant rapide, universel et précis. Nous devons maintenant analyser le travail qui a été fait. Voici les 11 éléments à l'étude:

- Modèle du canal de mesure;
- Filtre stationnaire ou non-stationnaire;
- Matlab versus C;
- Influence du pas d'intégration dans la méthode des trapèzes;
- Signal: forme, amplitude;
- Échantillonnage: pas, nombre de points;
- Influence de σ_ξ^2 et σ_η^2 (ou β);
- Effet de la contrainte de positivité: sans contrainte, contrainte dure et douce;
- Influence du bruit selon la variance pour une distribution normale;
- Calcul d'erreur de reconstitution;
- Temps de calcul.

Ces points sont discutés dans ce qui suit ainsi que dans les chapitres suivants.

Comme nous l'avons vu, nous avons développé plusieurs modèles d'état afin de simuler le canal de mesure. La table II montre l'erreur relative d'approximation du signal (εx) et de sa dérivée (εdx) selon le modèle utilisé, obtenue par simulation de l'équation (46):

$$\mathbf{v}_{n+1} = \mathbf{\Phi v}_n + \mathbf{b}\xi_n \tag{73}$$

connaissant $\mathbf{\Phi}$, $\mathbf{b}$ et ξ définis par le modèle utilisé.

Table II: Erreur de simulation selon le modèle.

Modèle	εx	εdx
Ordre 1	0.4825	0
Ordre 2	0.2358	0.4227
Ordre 1 continu	0.0111	0

IMPORTANT Nous voyons que le modèle d'ordre 1 donne de meilleurs résultats et que la continuité améliore encore plus le résultat. Les résultats des chapitres suivants utilisent le modèle d'ordre 1 (sauf si précisé autrement) puisque tous les modèles ont été étudiés et que ce dernier donnait les meilleurs résultats de reconstitution. En plus d'un choix de modèles, nous pouvons choisir entre le filtre stationnaire ou non-stationnaire comme décrit auparavant.

Nous nous apercevons rapidement que le traitement stationnaire est plus rapide et plus exact parce que le filtre stationnaire n'a pas à faire évoluer son vecteur de gain de Kalman vers des valeurs optimales. En effet, dans ce filtre, nous utilisons $\mathbf{k}_\infty$. De plus, il est plus rapide car nous n'avons pas à recalculer ce gain pour chaque échantillon; nous reprenons toujours le même. Finalement, puisque la fonction de transfert d'un instrument de mesure une fois stabilisée et prêt a être utilisée, varie rarement par rapport au temps, l'utilisation du filtre non-stationnaire semble superflu ou du moins, surement plus complexe. Une fois l'algorithme stationnaire optimisé en Matlab, il a été implanté en langage C pour se prévaloir de la vitesse de traitement que ce langage compilé peut obtenir.

Voici encore un point important. Peu importe l'ordre ou le modèle utilisé ou que le filtre soit stationnaire ou non, il faut toujours calculer le vecteur **h**. Selon (42), ceci demande une intégration. Ce type de calcul peut être réalisé numériquement, (nous ne connaissons que rarement l'expression de g) selon différents algorithmes. Le plus simple est la méthode du rectangle. La précision donnée par cette méthode est malheureusement trop imprécise pour obtenir une bonne reconstitution. La méthode des trapèzes donne de bons résultats. La méthode de Simpson ralentit la vitesse du traitement et augmente la complexité du calcul.

L'algorithme en langage C utilise donc la méthode des trapèzes pour faire l'intégration du calcul du vecteur **h**. Cela peut être une source importante d'inexactitude des résultats de reconstitution lors de l'application de la méthode étudiée dans ce mémoire.

VII. Implantation sur DSP56001

L'étape finale a été d'implanter l'algorithme dans un processeur; soit le DSP 56001 de Motorola afin d'identifier les contraintes d'implémentation inhérentes aux applications réelles. Etant donné l'espace mémoire requis et la vitesse de traitement désirée, la méthode implantée est stationnaire d'ordre 1 avec contrainte de positivité (SplKal+ 1). De plus, étant donné qu'un système de mesure intégré doit être dédié à un certain type de mesures pour être efficace, la fonction de transfert sera toujours la même et le niveau de bruit sera souvent de même ordre; de sorte que le vecteur $\mathbf{h}$ et le vecteur de gain de Kalman $\mathbf{k}_\infty$ peuvent être précalculés dans un ordinateur puissant et chargé ultérieurement dans le processeur. Une librairie de vecteurs $\mathbf{h}$ et $\mathbf{k}_\infty$ pourrait aussi être créée afin de permettre une certaine versatilité. Le processeur n'applique donc que la formule (66) ou Φ est traité comme une translation suivie de l'application des équations décrivant le modèle. Aussi, l'algorithme ne fut pas développé pour fonctionner en temps réel, ceci permet un filtrage du résultat par le filtre de Kalman et améliore la reconstitution.

À cause de la représentation fractionnaire des nombres dans le DSP56001, toutes les variables devaient être contenues dans l'intervalle [-1.0, 0.9999998] afin de pouvoir être sauvegardées en mémoire. En règle générale, les vecteurs $\tilde{\mathbf{y}}$, $\mathbf{h}$ and $\mathbf{k}_\infty$ contiennent des éléments qui ne répondent pas à cette limitation. En conséquence, ils ont dû être normalisés, premièrement le vecteur $\mathbf{h}$, et ensuite le gain de Kalman $\mathbf{k}_\infty$, en utilisant le vecteur $\mathbf{h}$ normalisé. À cause de cette limitation, la normalisation n'a pas pu être faite par le processeur DSP56001. Le pré-traitement ainsi que le post-traitement des données ont donc été accomplis par un processeur externe; soit un compatible IBM en MATLAB.

Le but de cette implantation en DSP est de montrer la précision pouvant être obtenue par ce processeur malgré ses limitations. Aussi, cette implantation permettra d'évaluer la vitesse de traitement possible, dans une application réelle, pour l'algorithme étudié.

VIII. Résultats

Donnée synthétique :

Beaucoup de simulations ont été faites avec différents paramètres, différentes formes de signaux et utilisant ou non une contrainte de positivité. Afin de mieux comparer nos résultats avec des résultats obtenus par d'autres types d'algorithmes, les principaux signaux étudiés ici seront ceux utilisés par Crilly [1]. Le signal d'entrée et le transopérateur sont définis par:

$$x(\lambda) = 2\exp(-6(\lambda - 5.5)^2) + 6\exp(-6(\lambda - 8)^2) \tag{74}$$

$$g(\lambda) = \begin{cases} 6(\exp(-0.7\lambda) - \exp(-0.9\lambda)) & \text{pour} \quad \lambda \geq 0 \\ 0 & \text{pour} \quad \lambda < 0 \end{cases} \tag{75}$$

et bien sûr on obtient les signaux de sortie non-bruité et bruité par:

$$y(\lambda) = g(\lambda) * x(\lambda) \tag{76}$$

$$\tilde{y}_n = y(\lambda_n) + \eta_n \qquad \text{pour } n = 0,1, \ldots, N \tag{77}$$

La figure 7 montre les deux pointes définissant le signal d'entrée (x) et ,en pointillé, le signal de sortie (y). Dans l'encart on retrouve la fonction de transfert ou transopérateur (g).

Ces signaux représentent un exemple commun dans le domaine de la spectrométrie. En effet, si une source de lumière entre dans un spectromètre, la lumière sera décomposée en ses couleurs pour former des raies ayant chacune une fréquence et une amplitude différentes. Mais la lumière de chaque raie se diffuse dans l'air avant d'atteindre le capteur et donc la raie s'élargit. Finalement, le capteur ou un mauvais alignement des lentilles pourrait aussi biaiser la forme.

Voyons aussi différents vecteurs utilisés à l'intérieur de l'algorithme pour faire la reconstitution précédente. La figure 8 montre l'allure du vecteur **h**. On reconnaît dans l'enveloppe de ce signal, la forme du transopérateur (g) comme on pouvait s'y attendre selon l'équation (42).

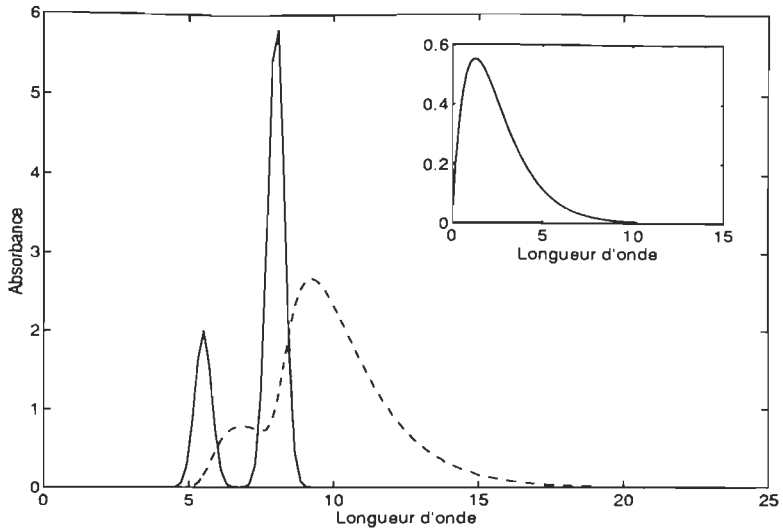


Fig 7 : Signaux de Crilly.

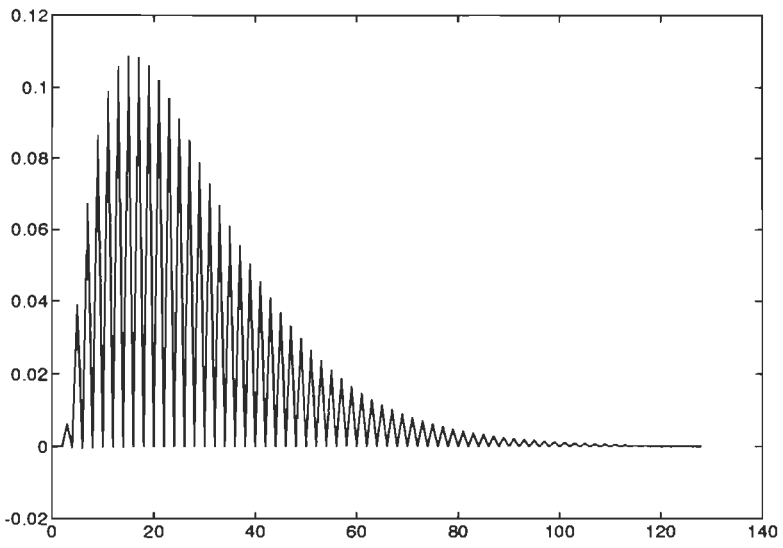


Fig 8 : Allure du vecteur **h** pour le transopérateur de Crilly.

La figure 9 montre l'allure du vecteur de gain de Kalman $\mathbf{k}_{\infty}$. À l'examen de cette courbe, on remarque aussitôt son allure oscillante anormale. La cause de ce phénomène est que le vecteur résultant du filtre de Kalman (**v**) ne contient pas uniquement les valeurs de x mais alternativement un x et un $\dot{x}$. La même conclusion aurait d'ailleurs pu être tirée de la figure 8.

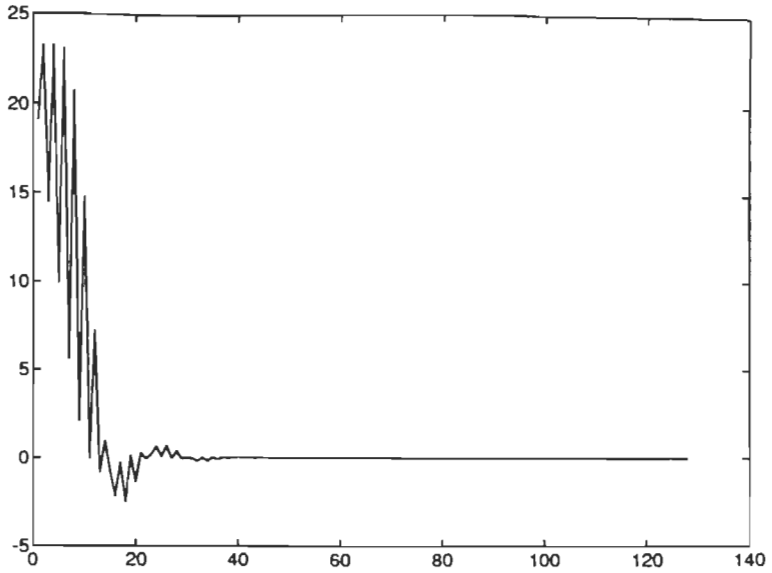


Fig 9 : Vecteur du Gain de Kalman calculé pour le transopérateur de Crilly; $\beta = 1000$.

Voyons un premier résultat de reconstitution. La figure 10 montre les différents signaux nécessaires à l'étude, soient le signal d'entrée, le transopérateur, le signal de sortie perturbé par un bruit de distribution normale ayant un écart type de 0.01 et le signal d'entrée reconstitué. Ce résultat a été obtenu avec l'algorithme stationnaire d'ordre 1 sans contrainte de positivité et avec β=1000. L'erreur relative de reconstitution est de 0.3678. De plus, cette figure montre le type de présentation graphique que donne le système SCR.

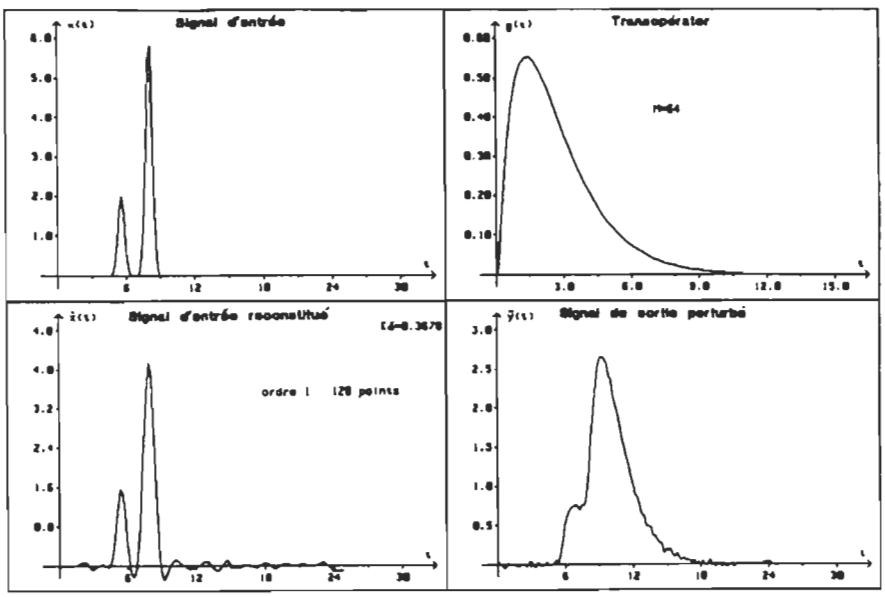


Fig 10 : Exemple de reconstitution non-optimale du signal de Crilly.

L'algorithme semble donc implanté sans erreur car le traitement par le filtre de Kalman, en utilisant les variances optimales, donne des résultats convenables. Mais quelle est exactement l'influence de la précision des variances données? Une étude a été faite afin de visualiser le phénomène. Pour chaque modèle décrit auparavant, des simulations ont été faites avec 15 ensembles de données bruitées au même niveau mais avec des réalisations de bruit différentes. Les figures 11, 12 et 13 montrent l'influence du paramètre β sur l'erreur de reconstitution pour ces simulations. Le paramètre β varie de 10^2 à 10^6. On cherche à voir l'étendue de valeur que β peut prendre tout en donnant une erreur minimale et aussi la répétitivité selon la réalisation. La figure 11 montre les résultats de cette étude, obtenus pour le modèle d'ordre 1; l'erreur minimale se situe entre 0.17 et 0.27 pour un β allant grosso-modo de 14^2 à 12^4. La figure 12 montre les résultats obtenus pour le modèle d'ordre 2; l'erreur minimal se situe entre 0.17 et 0.27 mais ici β va de 14^2 à 11^4. La figure 13 montre les résultats obtenus pour le modèle d'ordre 1 continu; l'erreur minimale se situe entre 0.31 et 0.37 et β va de 11^3 à 11^5. Cette dernière figure est plutôt chaotique et toute conclusion tirée d'elle serait douteuse. Les figures 11 et 12, tant qu'à elles, montrent des résultats similaires malgré la différence d'échelle. De plus, la translation des β optimaux entre les deux graphiques s'explique par la différence inhérente aux deux modèles et déja mentionnée afin d'aboutir aux équations (70) et (71).

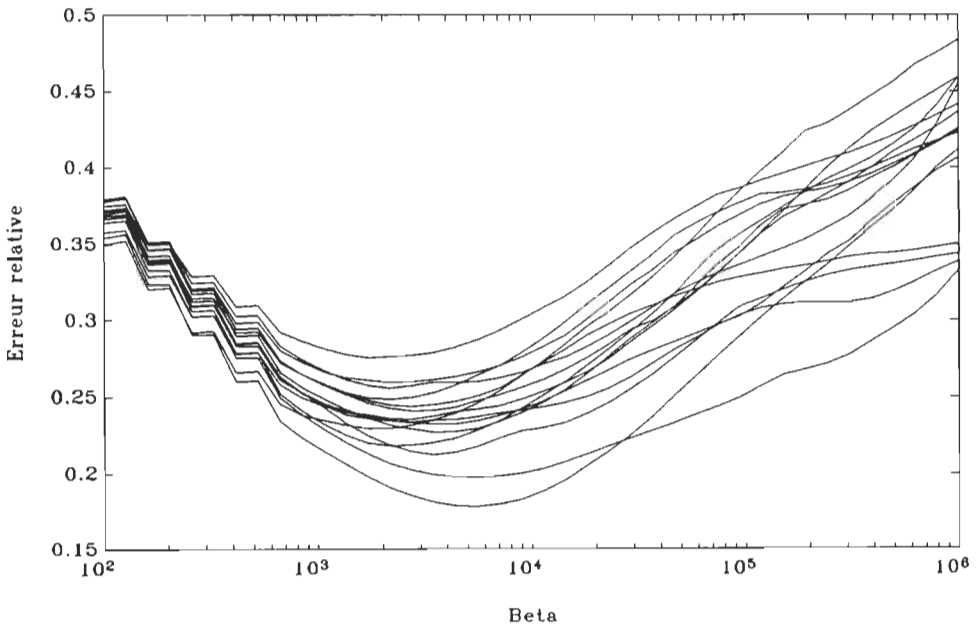


Fig 11: Influence de β pour 15 réalisations de bruit différents; filtre d'ordre 1.

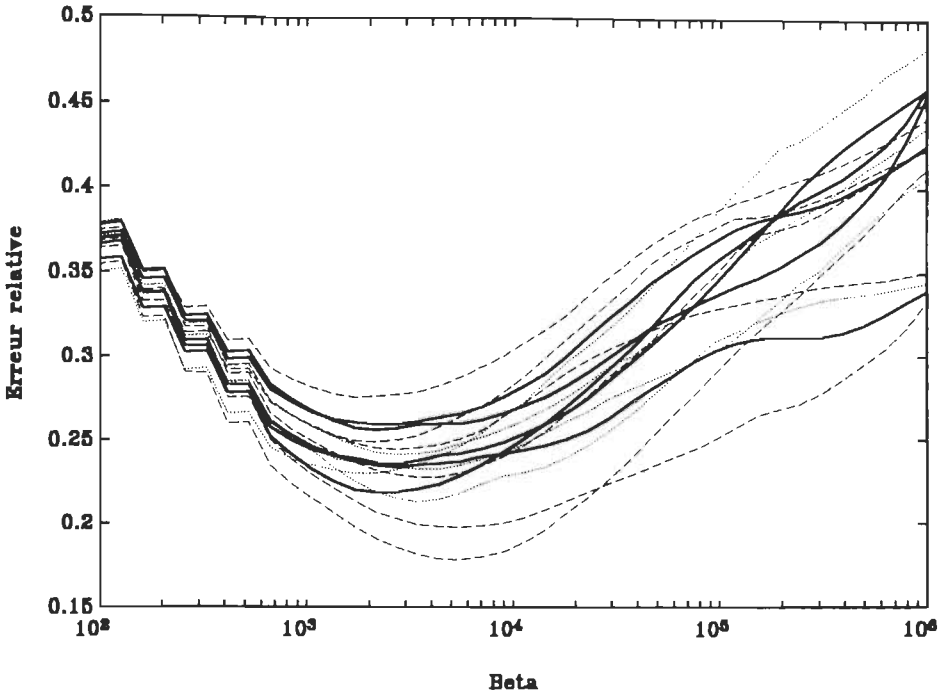


Fig 12 : Influence de β pour 15 réalisations de bruit différents; filtre d'ordre 2.

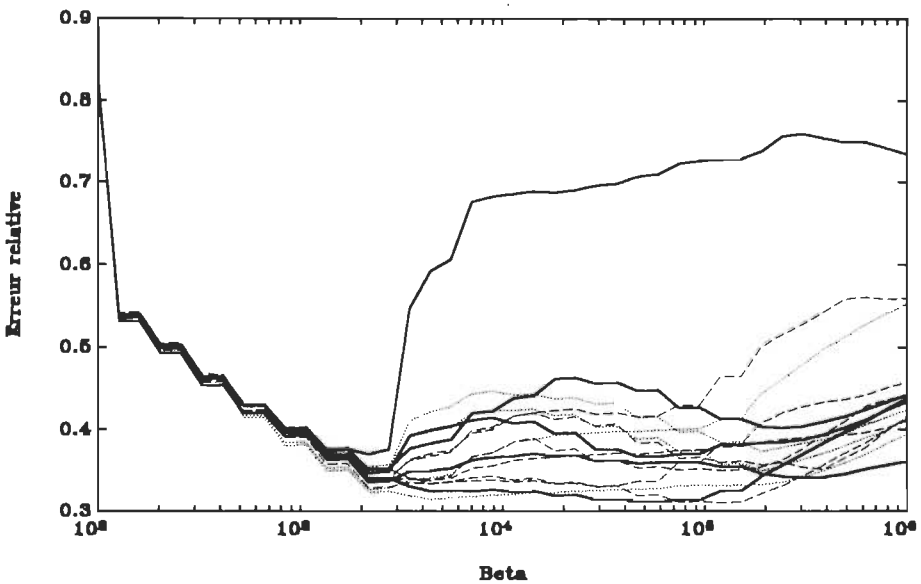


Fig 13 : Influence de β pour 15 réalisations de bruit différents; filtre continu.

Pour résumer les trois figures précédentes, on montre à la figure 14 l'erreur relative moyenne des 15 réalisations de signal bruité pour chaque modèle.

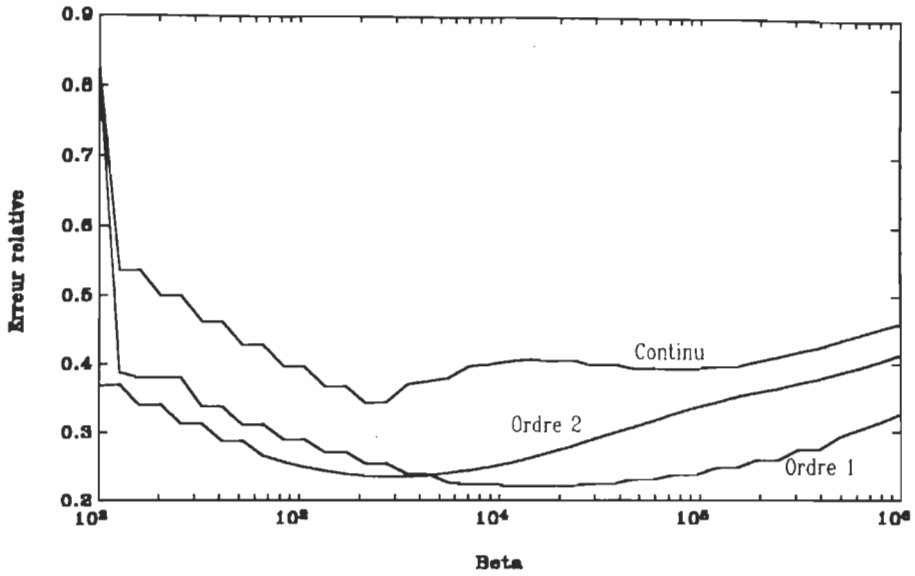


Fig 14 : Influence moyenne de β pour les trois modèles.

Outre le paramètre β, il y a le paramètre de la contrainte de positivité c qui a un grand rôle à jouer dans la présente étude algorithmique. Plus la valeur de ce paramètre s'approchera de zéro plus la contrainte sera grande. Une valeur de c = 1 n'occasionne aucune contrainte et une valeur de c = 0 occasionne une contrainte très forte appelée contrainte dure. Les valeurs intermédiaires occasionnent pour leur part une contrainte douce. Il est important d'optimiser ce paramètre car il peut beaucoup améliorer un résultat. Normalement, plus la contrainte est forte plus le résultat est bon. Mais il faut être prudent car une contrainte trop forte ou dure peut à ce point transformer le signal en traitement que l'algorithme ne le reconnaîtra plus. Puisque le filtre de Kalman est un processus causal, ce qui est passé dans le filtre a encore de l'importance pour traiter ce qui suivra. Si le passé est trop changé par la contrainte de positivité, le filtre traitera mal le présent! La figure 15 montre l'effet d'une trop forte contrainte. La pointe autour de β = 11^4 montre que la contrainte a empêché d'atteindre une erreur de reconstitution faible, normalement atteinte même sans contrainte.

En règle générale, un système ayant une fonction de transfert de forme symétrique, tel une gaussoïde, n'acceptera que très peu de contrainte (c=0.9); tandis qu'un système ayant une fonction de transfert de forme asymétrique acceptera plus de contrainte (c=0.3).

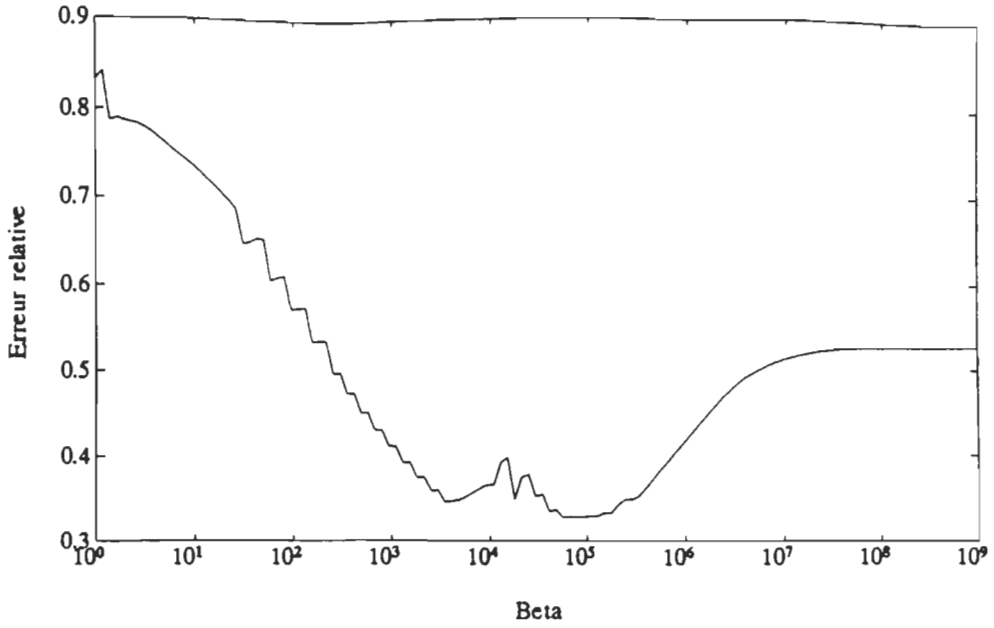


Fig 15 : Effet d'une trop forte contrainte de positivité. (c = 0.1)

Voyons maintenant l'efficacité de l'algorithme pour traiter des signaux ayant différents niveaux de bruit. Les mêmes signaux de Crilly sont utilisés mais avec des bruits additifs de distribution normale et d'écat type (σ_η) de 0, 1e-6, 1e-4 et de 1e-2. Les résultats pour le filtre d'ordre 1 sont montrés, ainsi que les paramètres utilisés, pour chaque écart type aux figures 16, 17, 18 et 19 respectivement. Une deuxième série de résultats pour le filtre d'ordre 2 sont montrés, avec les paramètres utilisés, pour chaque écart type aux figures 20, 21, 22 et 23 respectivement.

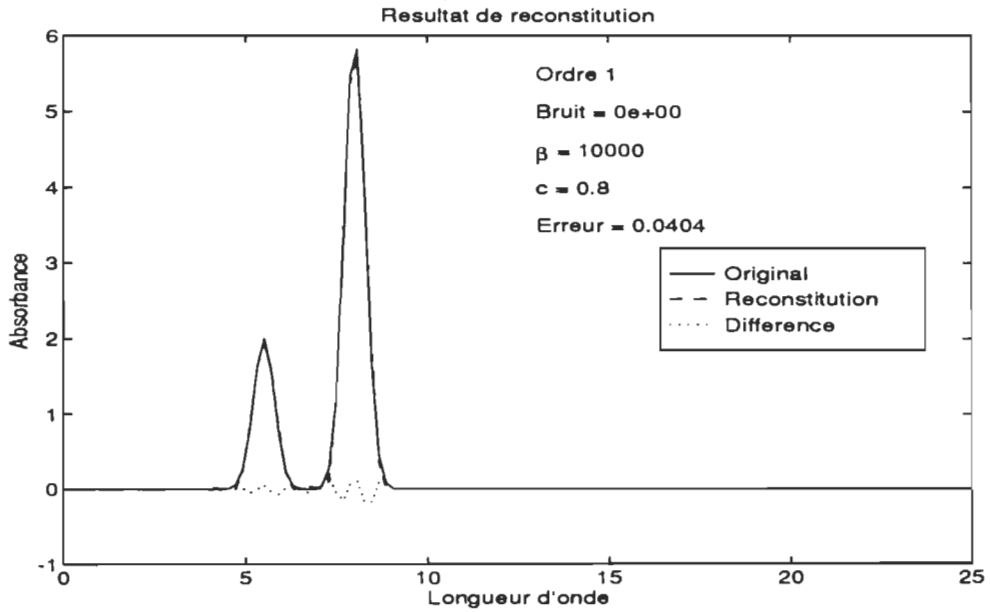


Fig 16 : Résultat de reconstitution optimale pour le filtre d'ordre 1 avec β = 1e4, c = 0.8 et l'écart type de bruit σ_η = 0.

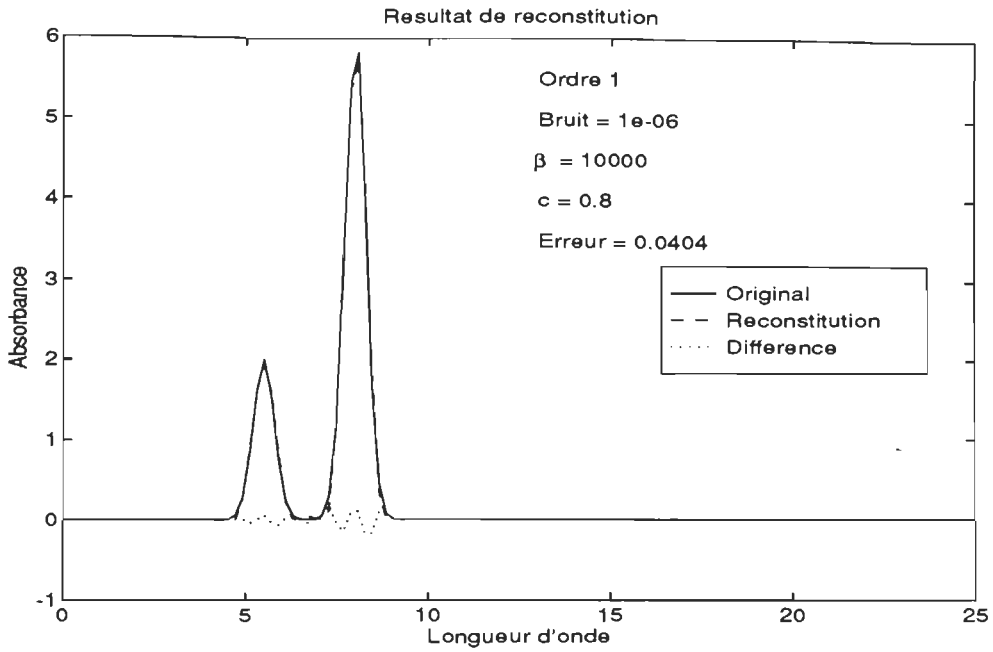


Fig 17 : Résultat de reconstitution optimale pour le filtre d'ordre 1 avec β = 1e4, c = 0.8 et l'écart type de bruit σ_η = 1e-6.

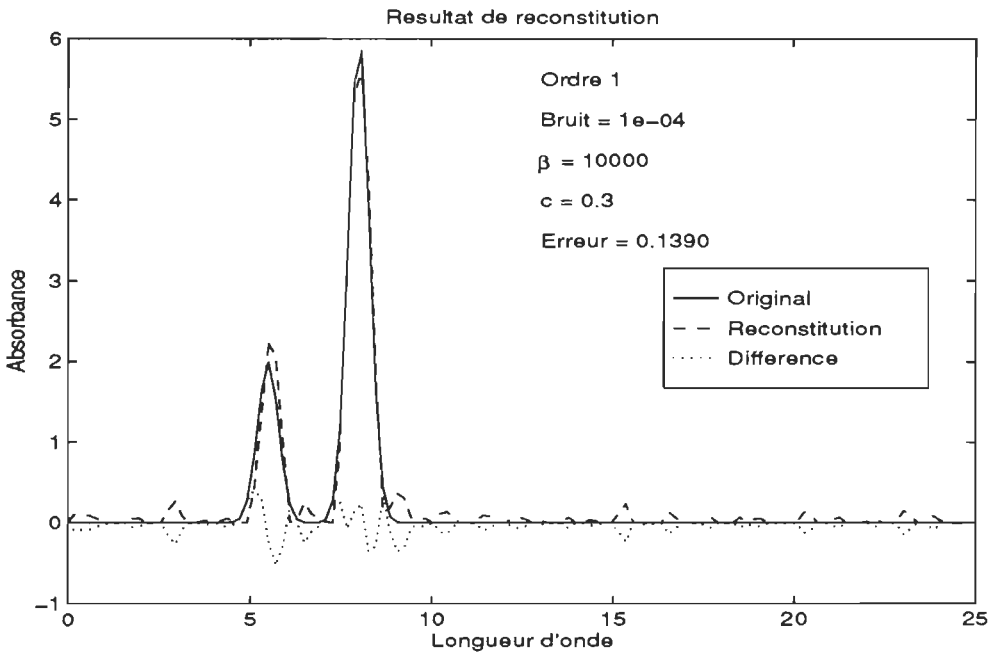


Fig 18 : Résultat de reconstitution optimale pour le filtre d'ordre 1 avec β = 1e4, c = 0.3 et l'écart type de bruit σ_η = 1e-4.

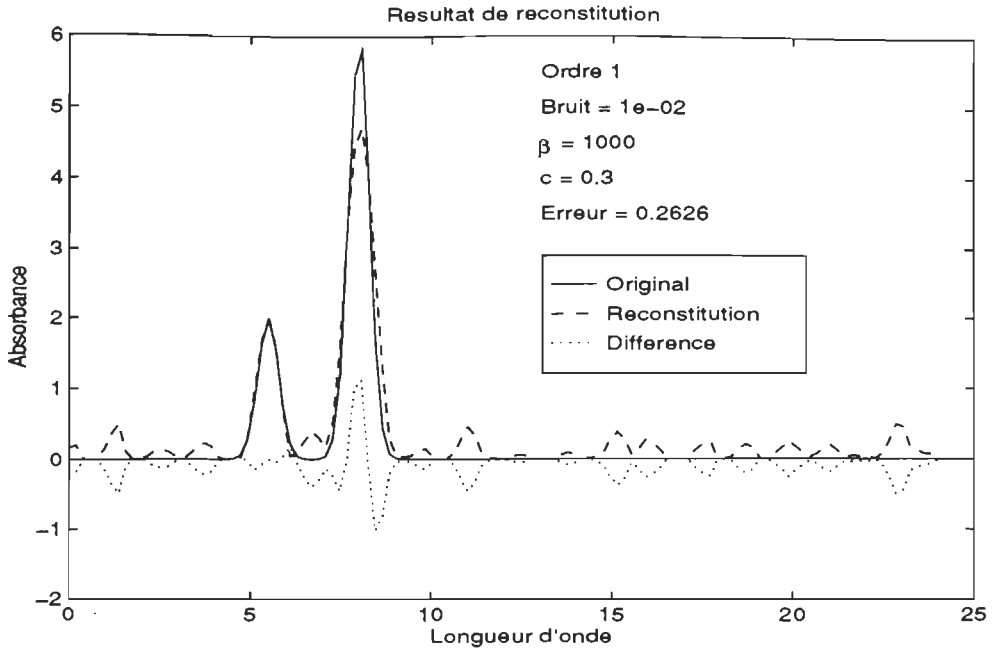


Fig 19 : Résultat de reconstitution optimale pour le filtre d'ordre 1 avec β = 1e3, c = 0.3 et l'écart type de bruit σ_η = 1e-2.

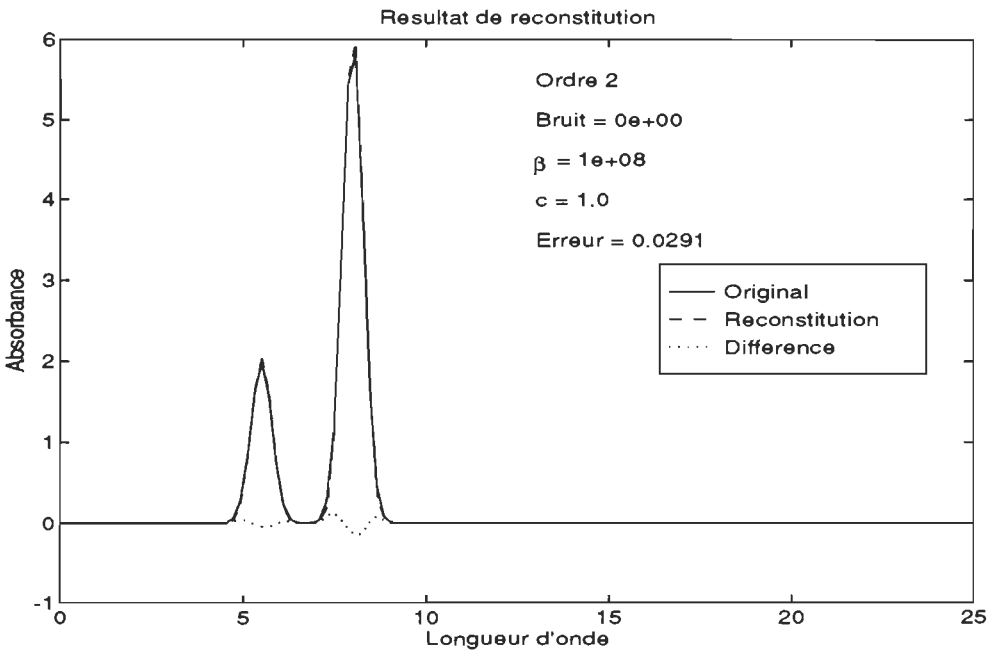


Fig 20 : Résultat de reconstitution optimale pour le filtre d'ordre 2 avec β = 1e8, c = 1.0 et l'écart type de bruit σ_η = 0.

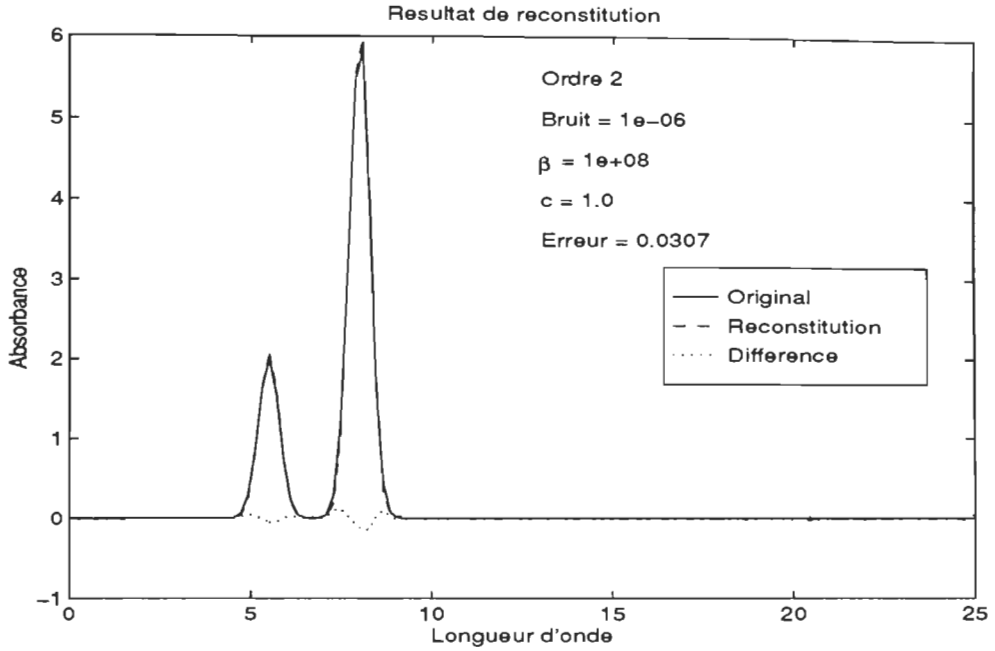


Fig 21 : Résultat de reconstitution optimale pour le filtre d'ordre 2 avec β = 1e8, c = 1.0 et l'écart type de bruit σ_η = 1e-6.

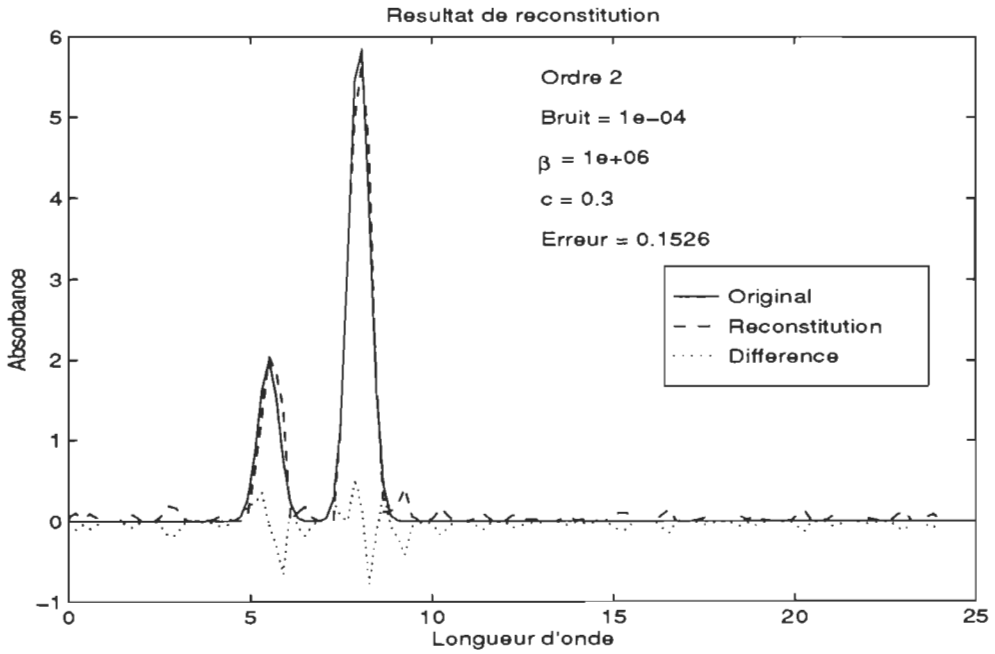


Fig 22 : Résultat de reconstitution optimale pour le filtre d'ordre 2 avec β = 1e6, c = 0.3 et l'écart type de bruit σ_η = 1e-4.

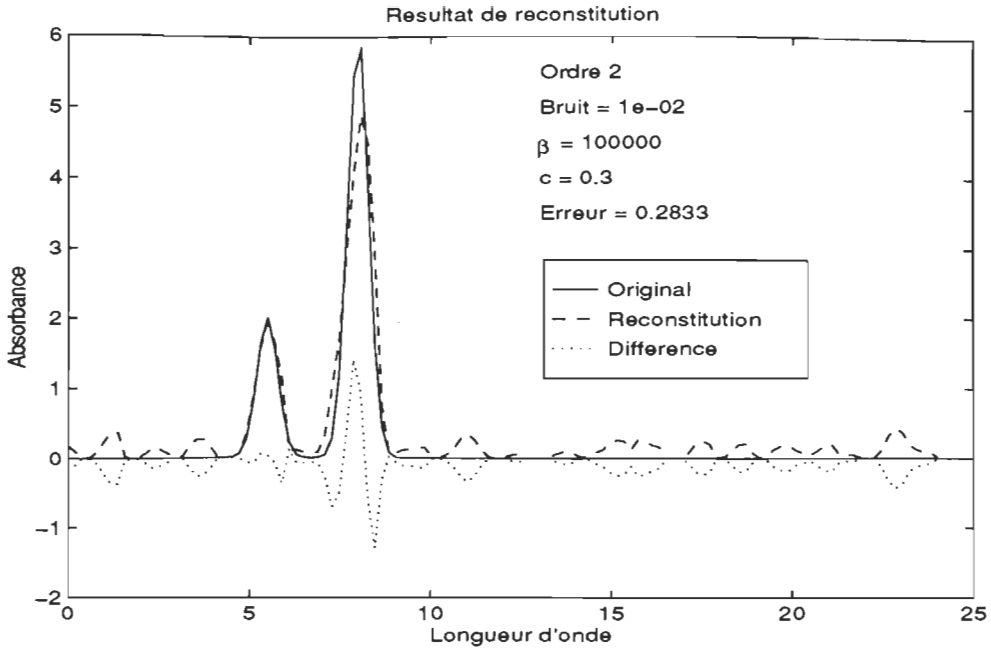


Fig 23 : Résultat de reconstitution optimale pour le filtre d'ordre 2 avec β = 1e5, c = 0.3 et l'écart type de bruit σ_η = 1e-2.

Nous avons vu comment le signal d'entrée peut être reconstitué par l'algorithme utilisant différents paramètres mais qu'en est-il de la première dérivée de ses signaux? Rappellons que cet algorithme prend sa force dans le fait qu'il considère la dérivée du signal dans son traitement. Voyons comment la dérivée est reconstituée par l'algorithme. Les figures 24 et 25 montrent la comparaison graphique entre le signal d'entrée et sa dérivée (trait plein) avec le signal reconstitué et sa dérivée reconstituée (trait pointillé). Et ce, pour le filtre d'ordre 1 et d'ordre 2 respectivement, chacun avec l'application d'une contrainte de positivité douce.

Évidemment, la contrainte de positivité n'est pas appliquée aux éléments du vecteur **v** qui donnent la dérivée du signal comme le demandait l'équation (67). L'application de la contrainte de positivité demande un réajustement des dérivées à chaque pas de calcul. La dérivée ajustée n'est donc qu'une approximation calculée à chaque pas!

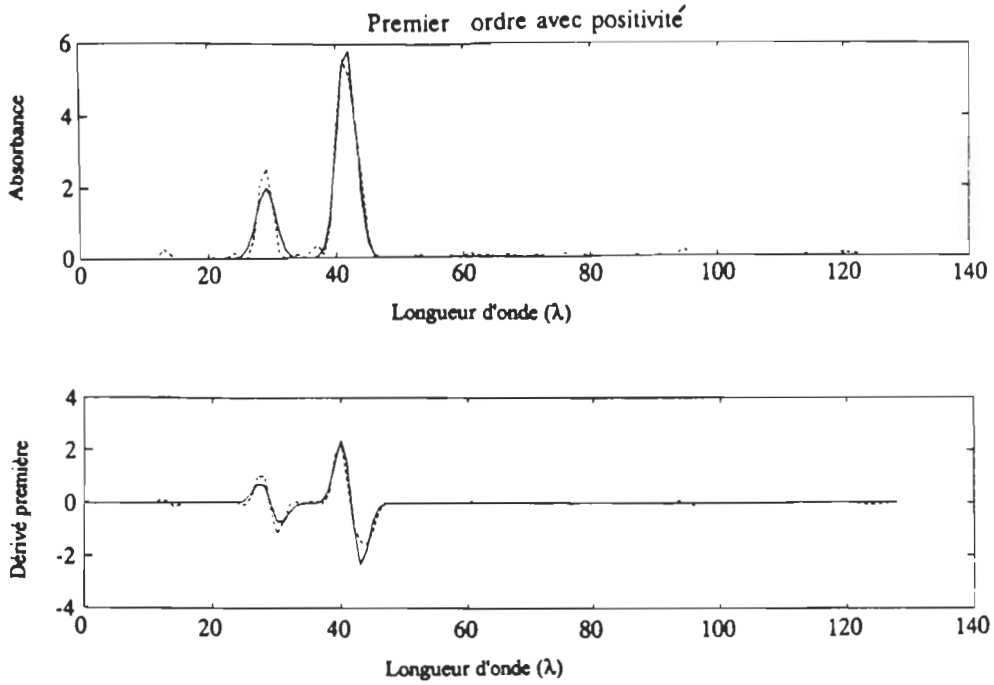


Fig 24 : Résultat de reconstitution de la dérivée pour le filtre d'ordre 1; erreur sur le signal égalant 0.1602, erreur sur la dérivée égalant 0.3393.

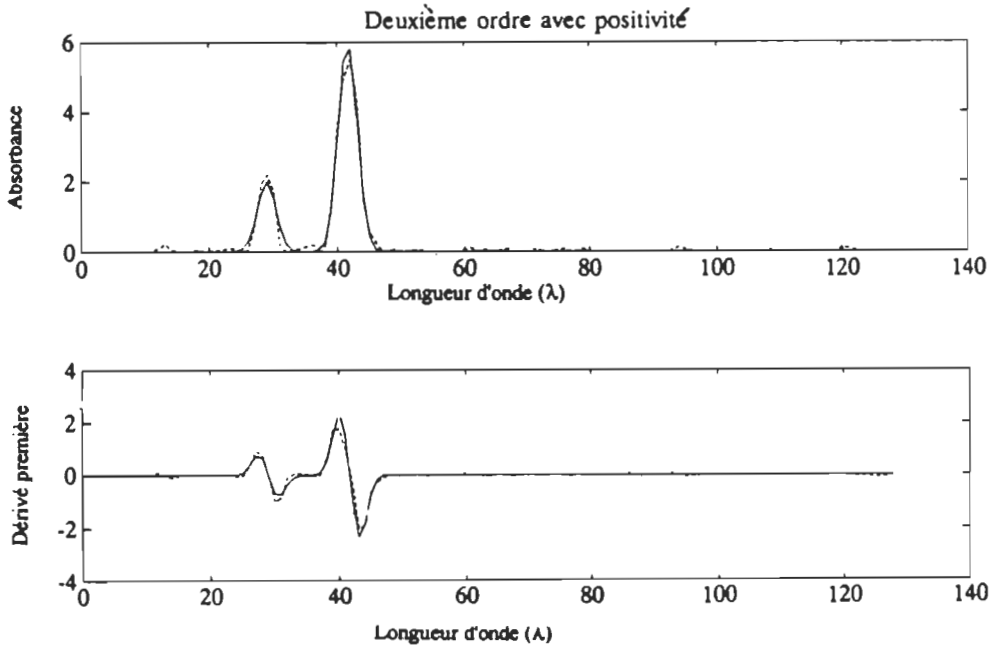


Fig 25 : Résultat de reconstitution de la dérivée pour le filtre d'ordre 2; erreur sur le signal égalant 0.1284, erreur sur la dérivée égalant 0.2660.

Finalement, ces mêmes signaux ont été traités par l'implantation DSP. La table III montre l'exactitude des résultats obtenus par DSP comparé à Matlab avec 10 réalisations

de bruit . Ici l'exactitude a été calculée par:

$$\bar{\delta}_2 = \sqrt{\frac{1}{10}\sum_{i=1}^{10}\delta_{2,i}^2} \tag{77}$$

ou $$\delta_{2,i}^2 = \frac{1}{N}\sum_{n=1}^{N}\left(\hat{x}_{n,i} - x_n\right)^2 \tag{78}$$

$\{\hat{x}_{n,i}\}$ - résultat de reconstitution obtenu pour la $i^{ième}$ réalisation.

Table III : Comparaison de l'exactitude de Matlab vs DSP pour N=128 et 256 avec σ_η = 0, 1e-6, 1e-4 et 1e-2.

N	Implémentation	σ_η			
		1e-6	1e-4	1e-3	1e-2
128	MATLAB	0.0024	0.0027	0.0172	0.2689
	DSP	0.0067	0.0070	0.0185	0.2725
256	MATLAB	8.88e-5	0.0016	0.0202	0.4424
	DSP	0.0014	0.0028	0.0216	0.4775

Pour avoir une idée de la vitesse de traitement par le processeur, la figure 26 montre des temps d'exécution en fontion de N (nombre de point de y) et K (nombre de point de g).

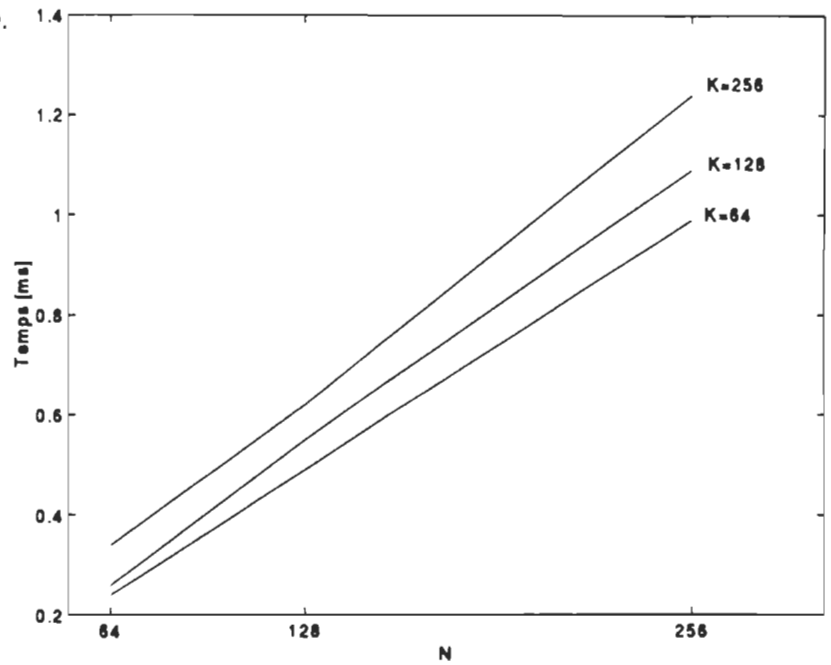


Fig 26 : Vitesse de traitement du DSP56001 selon le nombre de points N et K.

Donnée réelle :

Les signaux utilisés jusqu'à présent ont été crées par des formules mathématiques; ils sont utiles car ainsi on connaît à l'avance le résultat recherché (x). Ainsi, une erreur a pu être calculée pour fins de comparaison. Il serait intéressant de voir comment réagit l'algorithme à des signaux provenant de la réalité; non pas créés mathématiquement mais acquis sur un instrument de mesure réel.

Les signaux réels utilisés dans les figures qui suivent ont été acquis sur un spectromètre optique ANRITSU de la série MV02. Il était excité par deux lasers de précision; premièrement ajustés à des longueurs d'onde de 1537.7 et de 1540.0 nm et par la suite ajustés à des longueurs d'onde de 1537.7 et de 1539.1 nm et finalement n'utilisant qu'un seul laser ajusté à une longueur d'onde de 1537.7 nm. Les spectrogrammes suivants ont été enregistrés:

$\{x_{0.1,1,n} / n = 1, ...,128\}$ -Premier spectrogramme acquis avec une résolution de 0.1 nm;

$\{\tilde{y}_{1,1,n} / n = 1, ...,128\}$ -Premier spectrogramme acquis avec une résolution de 1 nm;

$\{\tilde{y}_{2,1,n} / n = 1, ...,128\}$ -Premier spectrogramme acquis avec une résolution de 2 nm;

$\{\tilde{y}_{5,1,n} / n = 1, ...,128\}$ -Premier spectrogramme acquis avec une résolution de 5 nm;

$\{x_{0.1,2,n} / n = 1, ...,128\}$ -Deuxième spectrogramme acquis avec une résolution de 0.1 nm

$\{\tilde{y}_{1,2,n} / n = 1, ...,128\}$ -Deuxième spectrogramme acquis avec une résolution de 1 nm;

$\{\tilde{y}_{2,2,n} / n = 1, ...,128\}$ -Deuxième spectrogramme acquis avec une résolution de 2 nm;

$\{\tilde{y}_{5,2,n} / n = 1, ...,128\}$ -Deuxième spectrogramme acquis avec une résolution de 5 nm;

$\{g_{1,k} / k = 1, ..., 64\}$ -Spectrogramme final acquis avec une résolution de 1 nm;

$\{g_{2,k} / k = 1, ..., 64\}$ -Spectrogramme final acquis avec une résolution de 2 nm;

$\{g_{5,k} / k = 1, ..., 64\}$ -Spectrogramme final acquis avec une résolution de 5 nm;

Un lissage a été apporté aux trois spectrogrammes finaux. Ces derniers on été utilisés comme fonction de transfert (g) pour faire la reconstitution des autres. Les premier et deuxième spectrogrammes acquis à une résolution de 0.1 nm vont servir de signal de référence (x) pour juger de la qualité de reconstitution. Les autres spectrogrammes acquis à des résolution de 1, 2 et 5 nm seront les signaux de mesure bruts ($\tilde{y}$) à traiter. Les figures 27, 28 et 29 montrent les résultats obtenus avec et sans contrainte de positivité pour la première série de spectrogrammes. Les figures 30, 31 et

32 montrent les résultats obtenus avec et sans contrainte de positivité pour la deuxième série de spectrogrammes.

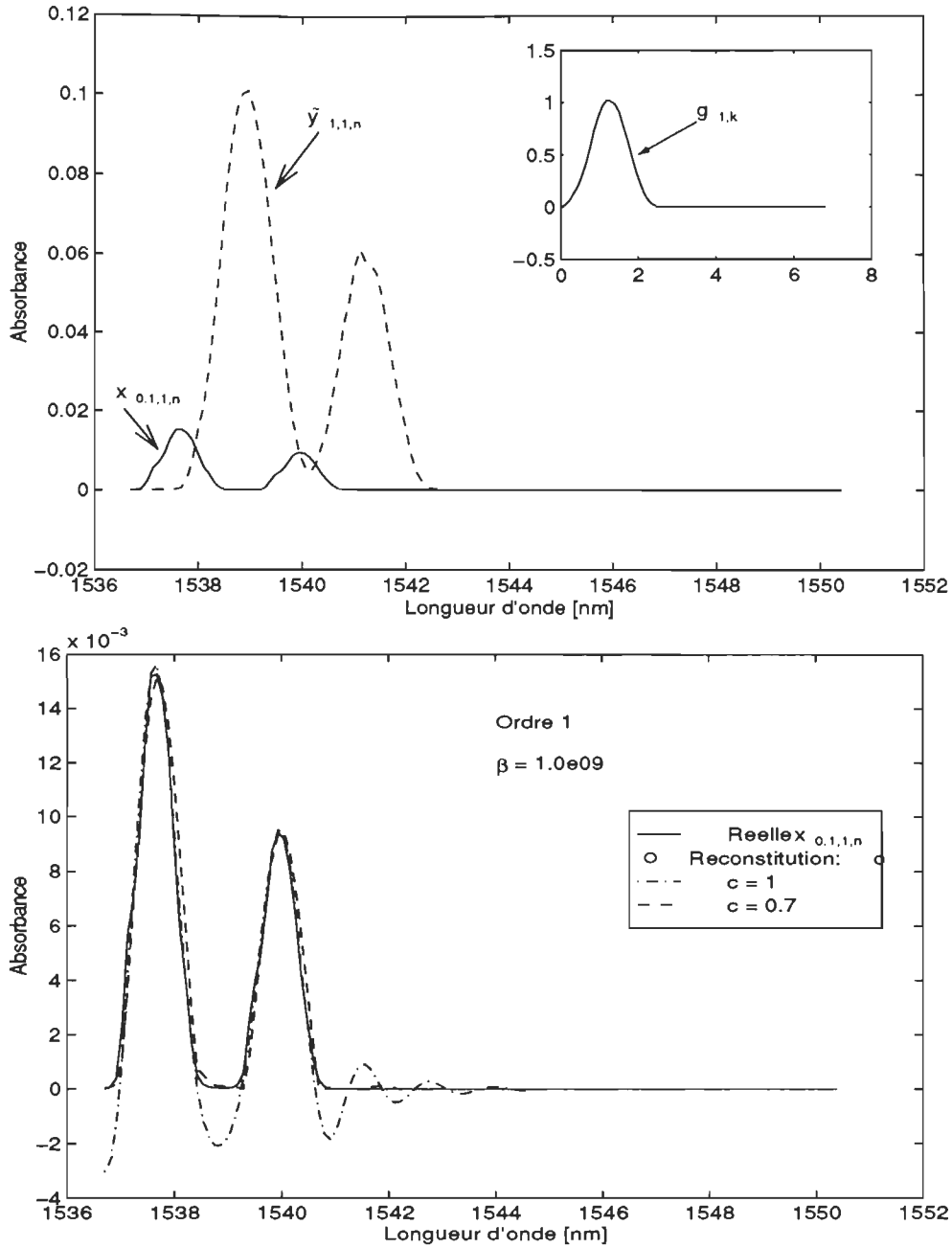


Fig 27 : Signaux de la première série et un résultat de reconstitution pour une résolution de 1 nm; β = 1e9 et c = 0.7.

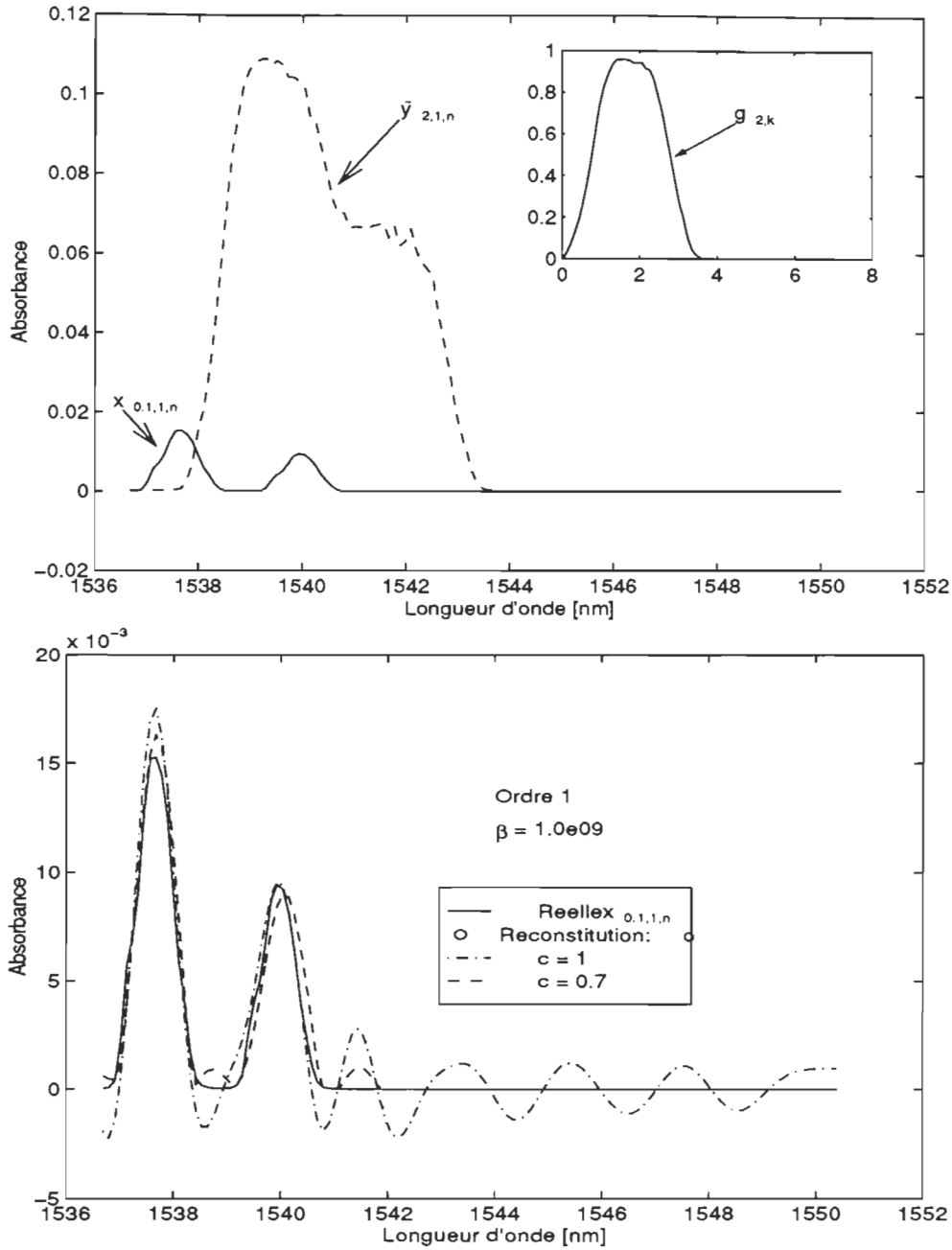


Fig 28 : Signaux de la première série et un résultat de reconstitution pour une résolution de 2 nm; $\beta = 1e9$ et $c = 0.7$.

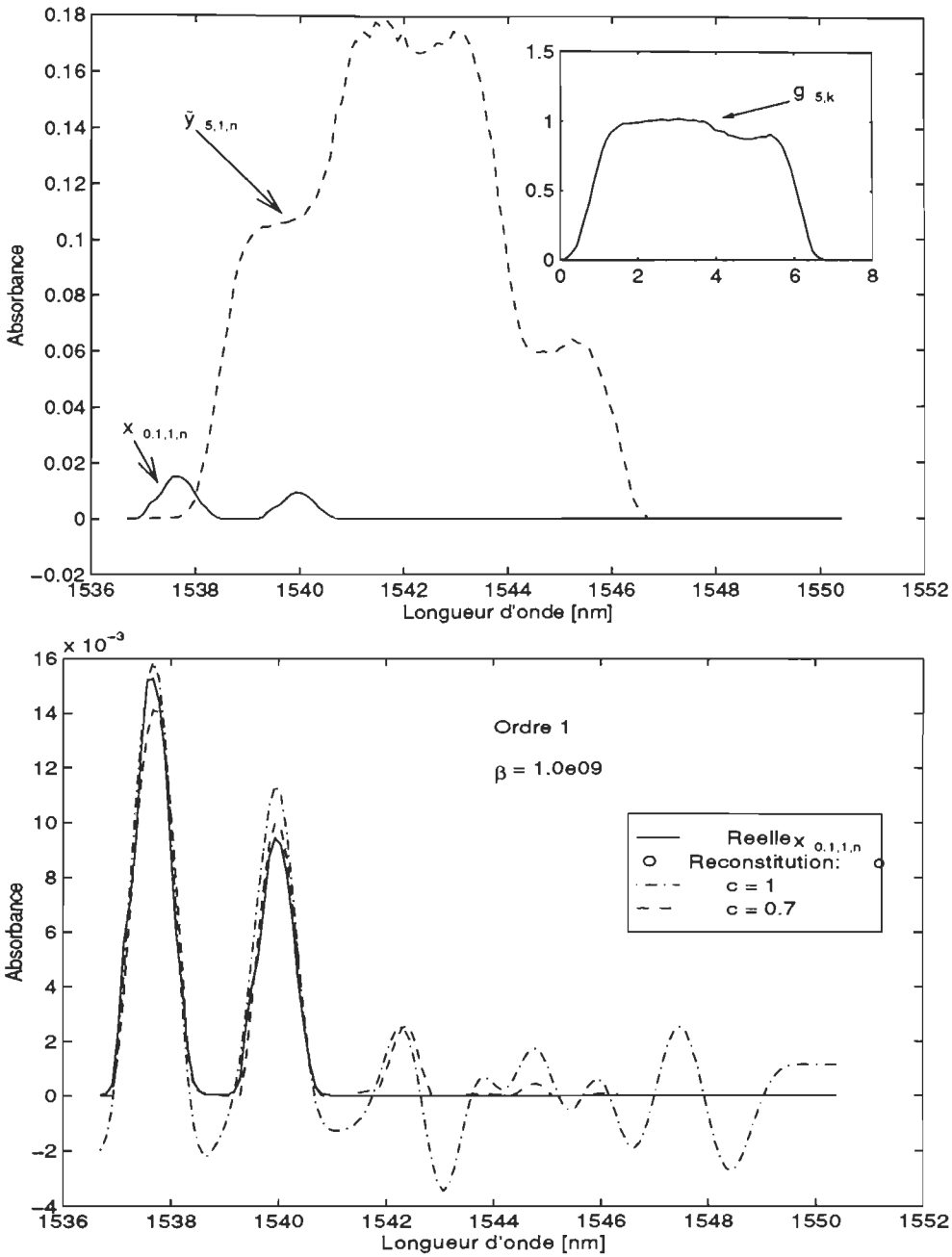


Fig 29 : Signaux de la première série et un résultat de reconstitution pour une résolution de 5 nm; β = 1e9 et c = 0.7.

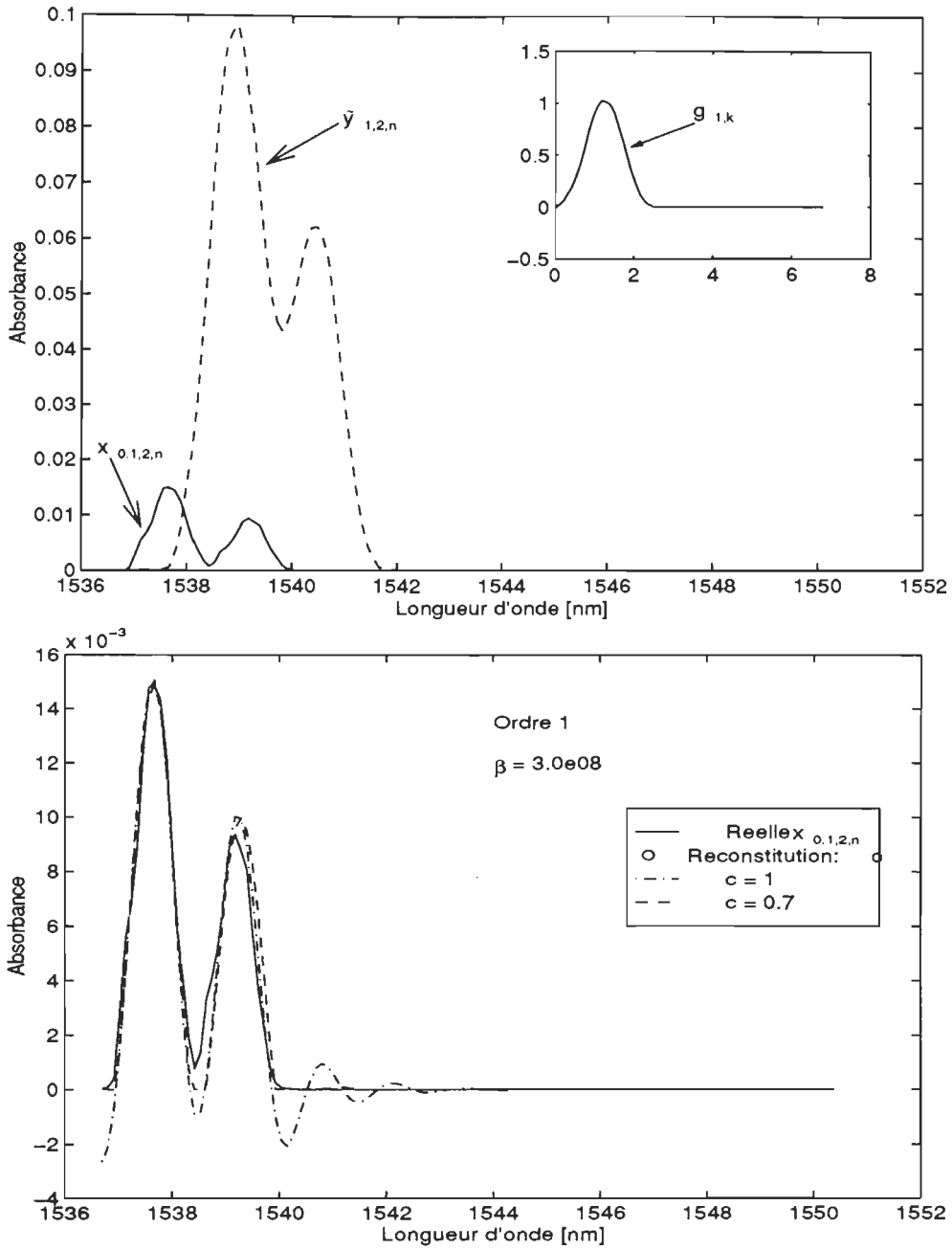


Fig 30 : Signaux de la deuxième série et un résultat de reconstitution pour une résolution de 1 nm; $\beta = 3e8$ et $c = 0.7$.

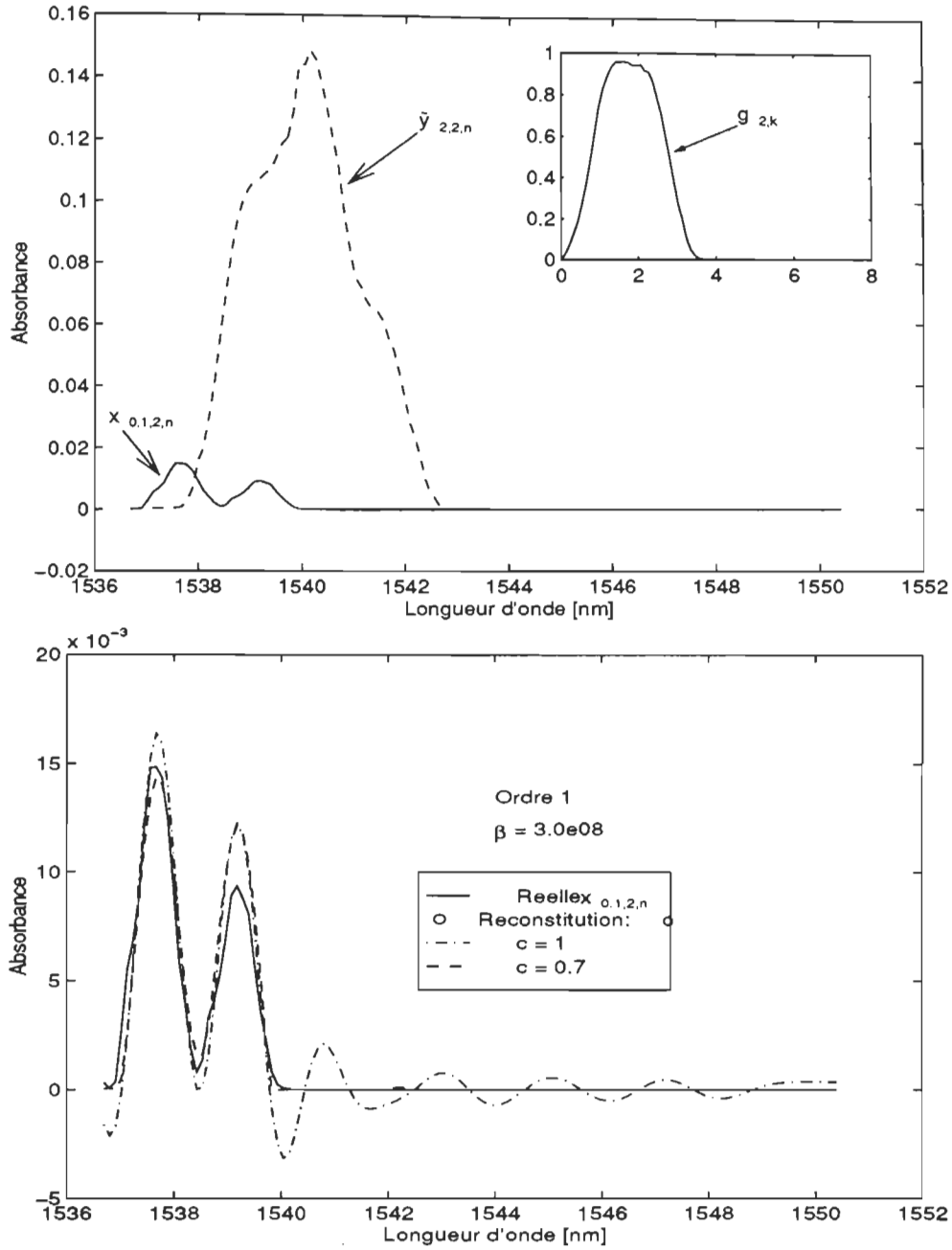


Fig 31 : Signaux de la deuxième série et un résultat de reconstitution pour une résolution de 2 nm; β = 3e8 et c = 0.7.

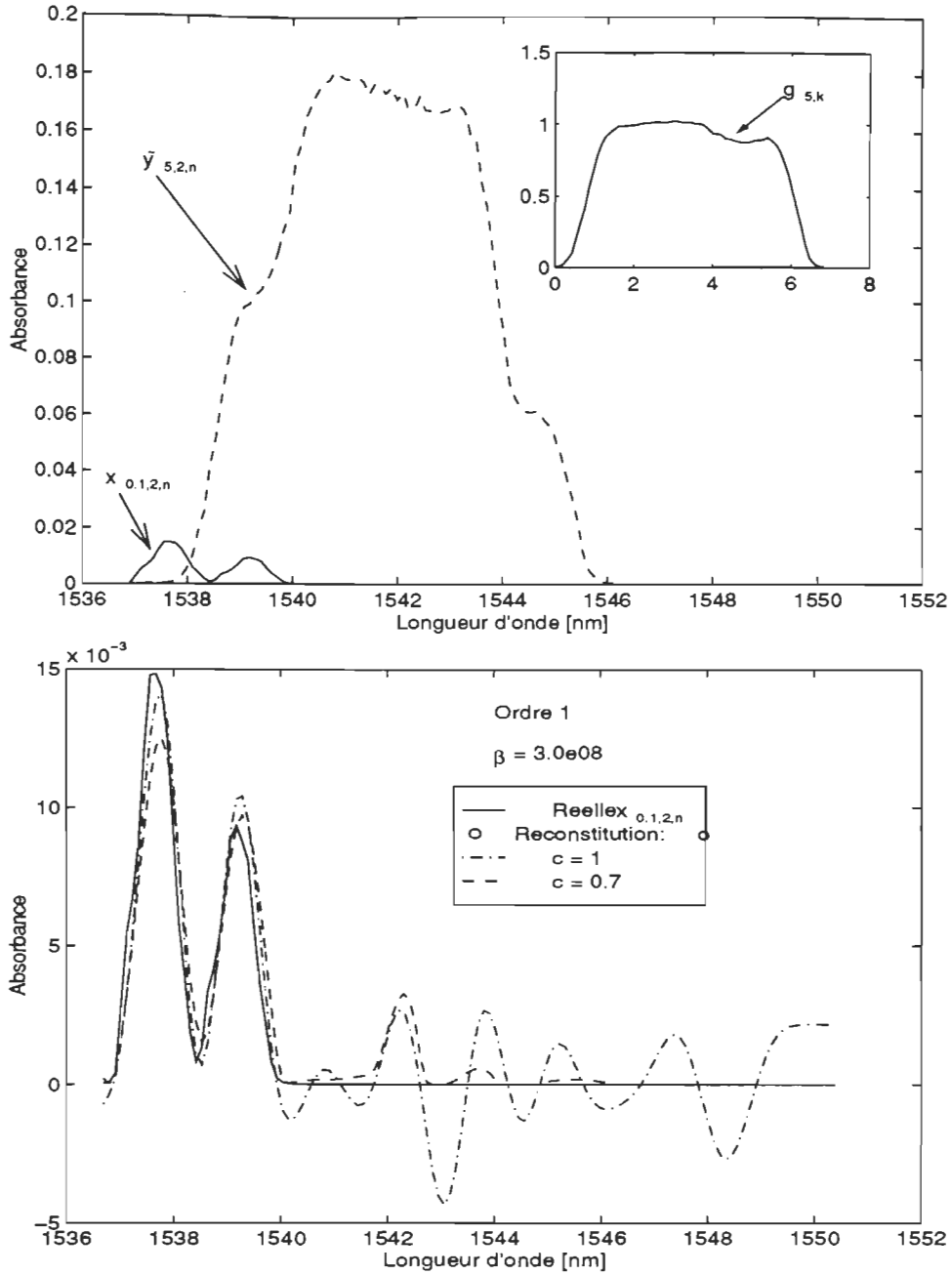


Fig 32 : Signaux de la deuxième série et un résultat de reconstitution pour une résolution de 5 nm; $\beta = 3e8$ et $c = 0.7$.

En règle générale, l'algorithme donne de bons résultats de reconstitution. Il est intéressant de noter que le paramètre β ne varie pas d'une reconstitution à l'autre malgré le changement de fonction de transfert et de signal d'entrée. Donc une fois étallonné ce paramètre n'a plus à être ajusté. Finalement, comme nous nous y attendions, l'ajout d'une contrainte de positivité douce améliore beaucoup les résultats.

IX. Discussion

Pour fins de comparaison la table IV montre l'erreur de reconstitution calculée selon (72) pour différentes méthodes et différents niveaux de bruit. On y présente des méthodes classiques telles que: Jansson et Van Cittert (méthodes itératives)[1] ainsi que Tikhonov (méthode de régularisation spectrale)[8]. La méthode appelée SplKal est la méthode décrite dans ce travail en mode stationnaire soit d'ordre 1, soit d'ordre 2. Le signe + dans le nom de la méthode indique l'utilisation ou non d'une contrainte de positivité. La méthode appelée SplKal EXP a été développée par un membre du laboratoire de système de mesure et demande de pouvoir représenter le transopérateur par une combinaison linéaire de fonctions exponentielles[9]. Finalement, la méthode appelée Kalman(+) ne fait appel qu'au filtre de Kalman et ne se soucie guère des dérivées du signal reconstitué[5].

Table IV : Comparaison d'erreur de reconstitution avec différentes méthodes pour les signaux de Crilly; N = 128, M = 64.

Méthode	Écart type du bruit σ_η			
	0	1e-6	1e-4	1e-2
Jansson	0.1578	0.1581	0.1721	0.5536
Van Cittert	0.7963	0.7962	0.8003	0.8834
Tikhonov	2.87e-4	0.0452	0.1670	0.4208
SplKal 1	0.0441	0.0449	0.2258	0.3881
SplKal 2	0.0291	0.0310	0.2768	0.4293
SplKal + 1	0.0404	0.0402	0.1584	0.2726
SplKal + 2	0.0291	0.0310	0.1512	0.3128
SplKal exp	0.0382	0.0410	0.1593	0.4616
Kalman	5.99e-5	0.0341	0.1374	0.4345
Kalman+	2.78e-5	0.0176	0.0764	0.2832

Rappellons finalement que toutes ces simulations ont été faites avec des signaux définis par 128 échantillons et un transopérateur défini par 64 points. Les résultats montrés pour Splkal 1, Splkal 2, Splkal + 1 et Splkal + 2 ont été optimisés en variant le paramètre β par décade entre 10^1 et 10^6 et le paramètre de contrainte c par dixième entre 0 et 1. Afin d'alléger la lecture de ce mémoire, les graphiques de surface créée par ces

calculs ont été mis en annexe. Néanmoins, la figure 33 montre une de ces surfaces pour le filtre d'ordre 2 et un niveau de bruit de σ_η=1e-2. On voit un minimum d'erreur en c=0.3 et β=1e5.

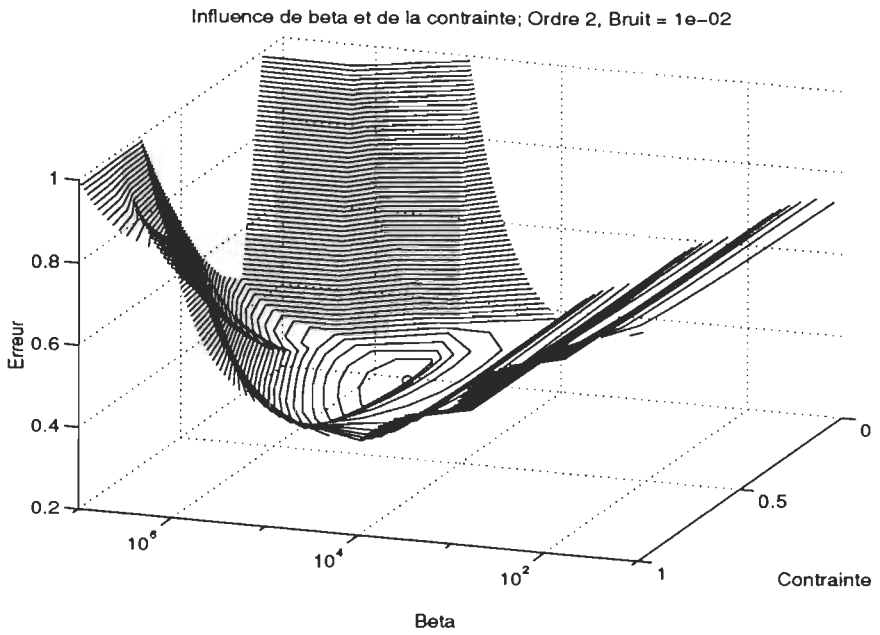


Fig 33 : Surface d'optimisation des paramètres β et c pour le filtre d'ordre 2 et σ_η = 1e-2.

Il est à noter que chaque signal a été bruité par 10 réalisations de bruit différentes. Donc les erreurs inscrites à la table IV (et montrées sur les surfaces), pour l'algorithme étudié dans ce mémoire, sont des erreurs moyennes. Comparées aux erreurs montrées sur les figures 16 à 23 qui ne sont qu'un exemple parmis les 10 réalisations, les erreurs de la table IV peuvent être légèrement plus petites ou plus grandes.

Les résultats montrés dans la table IV résument la qualité de reconstitution pouvant être obtenue par différents algorithmes en se basant sur l'erreur relative de reconstitution. Il faut mentionner que ces résultats ont été calculés par différentes personnes, sur différents ordinateurs. Chaque personne avait sa façon de voir et d'interpréter ses résultats. De plus, différents algorithmes demandent parfois des prétraitements ou des conditions particulières réduisant ainsi leurs champs d'action mais améliorant leurs performances.

Voici donc, pour bien établir les faits, les quelques particuliarités de l'algorithme étudié:

- le pas d'échantillonnage de la fonction de transfert doit être égal au pas du signal d'entrée;
- la fonction de tranfert doit être causale et ne peut donc pas être centrée à zéro;
- les signaux obtenus par des instruments non-causaux (spectromètre) doivent subir des translations sur l'axe horizontal pour compenser ce qui a été précisé au point précédent;
- comme déjà mentionné, le filtre stationnaire demande un précalcul du gain;
- le filtre non-stationnaire est lourd en calcul et donc lent d'exécution;
- la contrainte de positivité, même douce, rend l'algorithme instable si la fonction de transfert est symétrique;
- la continuité des dérivées apportée par spline rend discutable l'utilisation d'une contrainte de positivité.

Une dernière étude a été faite afin de comparer l'algorithme implanté en DSP avec un autre algorithme aussi implanté en DSP. La méthode spectrale avec régularisation de Tikhonov [8] a été utilisée. La table V montre, pour deux niveaux de bruit différents, l'exactitude de reconstitution des algorithmes implantés en DSP. Ici, N=128 et l'exactitude est calculée par les Eq. (77) et (78).

Table V : Comparaison d'erreur de reconstitution entre SplKal+ 1 sur DSP et Tikhonov sur DSP pour les signaux de Crilly; N = 128.

	σ_η	
Algorithme	1e-6	1e-2
SplKal+ 1	0.0067	0.2725
Tikhonov	0.0078	0.3934

X. Conclusion

Ce travail a été fait dans le cadre d'une recherche sur la possibilité d'intégration de systèmes de mesure. Maintenant que l'on maîtrise bien tous les éléments faisant partie de l'étude d'un algorithme de reconstitution, pour en avoir développé et étudié un, on peut très bien tirer quelques conclusions.

Les signaux de Crilly avec leur fonction de transfert non-symétrique permettent l'utilisation d'une contrainte de positivité. Pour d'autres types de signaux, par contre, cette contrainte peut créer une instabilité et une divergence du filtre de Kalman. Les fonctions de transfert des signaux réels sont suffisamment asymétriques pour permettre l'utilisation de la contrainte.

On a beaucoup à gagner en implantant le système de reconstitution de façon stationnaire: meilleure précision (voir table IV), moins de mémoire nécessaire (pas de Φ ,etc), moins de temps de traitement et simplicité d'implantation sur carte DSP entre autres ou d'intégration en VLSI.

Il est à noter que la précision obtenue lors du calcul du vecteur **h** est un facteur déterminant pour obtenir une bonne reconstitution. Ce vecteur est une collection d'intégrales, donc plus le calcul des intégrales sera précis plus les résultats seront bons. Pour améliorer le résultat d'une intégration numérique on doit avoir le plus d'échantillons possible. Mais qui dit plus de points dit plus de temps. Heureusement, pour un système stationnaire, les valeurs de **h** ne changent pas. On peut donc faire le précalcul de **h** une fois avec plusieurs points et toujours utiliser le même vecteur pour traiter différents niveaux de bruit ou différents signaux d'entrée; en autant que le transopérateur ne change pas.

L'utilisation des divers modèles présentés donne des résultats différents, mais visuellement semblables. La simplicité semble être la règle d'or lorsque nous développons un algorithme qui se veut efficace; filtre stationnaire, ordre un, contrainte douce, uniformité des pas d'échantillonnage parmi les signaux, et fonction de transfert complètement positive ...

L'algorithme proposé est très souple et demande peu de connaissances *a priori* sur les signaux à traiter. Entre autres, la précision demandée au paramètre β n'est pas très grande. Les résultats obtenus sont très intéressants surtout pour des niveaux de bruit élevés. Cet algorithme est supérieur aux méthodes plus classiques telles que Jansson et Tikhonov tout en ayant moins de complexité de calcul. Les méthodes basées uniquement sur spline sont très lentes et demandent beaucoup de mémoire. Pour cela, les signaux asymétriques de Crilly n'ont pas été étudiés avec ce type d'algorithme.[2] Les résultats obtenus ont été présentés à la conférence internationnale IEEE IMTC'94 [10] et la publication dans IEEE Transations on Instrumentation and Measurement est acceptée.

Dans la suite des traveaux, il serait intéressant de comparer la précision de l'algorithme dévéloppé avec celle offerte par l'algorithme Spline. L'algorithme proposé est probablement moins précis que les méthodes basées uniquement sur spline mais est tellement plus rapide; de plus l'algorithme étudié dans ce mémoire peut être implanté en DSP. D'autre par il serait également intéressant d'explorer d'avantage les possibilités d'améliorer l'algorithme développé afin d'obtenir une précision plus grande que celle offerte par l'algorithme Kalman+. En effet, l'ajout d'une contraite suplémentaire permet une réduction de l'ensemble de solutions possibles autour de la solution exacte.

Références

[1] P.B. Crilly, "A Quantitative Evaluation of Various Iterative Deconvolution Algorithms", IEEE Transactions on Instrumentation and Measurement, Vol. 40, pp. 558-562, June 1991.

[2] M. Ben Slima, R. Z. Morawski, A. Barwicz, "Spline-Based Variational Method with Constraints for Spectrophotometric Data Correction", IEEE Transactions on Instrumentation and Measurement, Vol. 41, pp. 786-790,Dec. 1992.

[3] Brian D. O. Anderson, John B. Moore, *Optimal Filtering*, Prentice-Hall, 1979.

[4] P. A. Jansson, *Deconvolution With Applications in Spectroscopy*, New York: Academic Press, ch. 1-4, pp. 1-134, 1984.

[5] D. Massicotte, R. Z. Morawski, A. Barwicz, "Incorporation of a Positivity Constraint Into a Kalman-Filter-Based Algorithm for Correction of Spectrometric Data", Instrumentation/Measurement Technology Conference (IMTC/92), New-York, Mai 1992, pp. 590-593.

[6] A. Miekina, A. Podgorski, M. S. Milewski, *The SCR System: Volume I*, Rapport interne, UQTR, pp.48-49,1992.

[7] L. Finkelstein, "Fundamental Concept of Measurement: Definitions and Scales", Measurement and Control, vol. 8, p.975.

[8] R. Z. Morawski, M. Ben Slima, M. S. Milewski, A. Barwicz, "Application of a Digital Signal Processor for Correction of Spectrophotometric Measurements", IEEE Transactions on Instrumentation & Measurement, Vol. 42, No. 3, June 1993, pp.778-782.

[9] M. Ben Slima, R. Z. Morawski, A. Barwicz, "A Recursive Spline-Based Algorithm for Spectrophotometric Data Correction", Conference Record IEEE IMTC'93, Irvine, California, May 1993, pp. 500-503.

[10] M. P. Brouard, R. Z. Morawski, A. Barwicz, "A DSP-Based Correction of Spectrogrammes using Cubic Spline and Kalman Filter", Conference Proceedings IEEE IMTC'94, Hamamatsu, Shisuoka, Japan, 10-12 May 1994, pp. 1443-1446.

Annexe:

Programme Matlab

Programme DSP56001

Surfaces d'optimisation de c et β pour les signaux du type Crilly

Publication: référence [10]

Caractéristiques du DSP56001 (photocopies)

Programme Matlab

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                     Kalman-Spline Method  statique                                  %
%                                                                                     %
%                                                                                     %
%              Author:    Pierre Brouard ,  UQTR,  7 avril 1994                       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% function [xr,hN,K]=SplKal(y,g,dt,B,c,hN,K,O);                                       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [x,hN,K]=SplKal(y,g,dt,B,c,hN,K,O);
Tstart=0;
M=length(g);
N=length(y);
if sum(K)==0                % Calcul du gain si non-fourni
  if sum(hN)==0             % Calcul de h si non-fourni
        du=dt/4;
        u=0:du:dt;
        t=(Tstart:dt:(M-1)*dt);
        W(1,:)=2*u.^3/dt^3-3*u.^2/dt^2+1;      % Définition des fonctions de contrainte spline
        W(2,:)=u.^3/dt^2-2*u.^2/dt+u;
        W(3,:)=-2*u.^3/dt^3+3*u.^2/dt^2;
        W(4,:)=u.^3/dt^2-u.^2/dt;
        HH=[0;0;0;0];
        for n=1:M-1                            % Intégration par trapèze avec 4 points interpollés
          H=spline(t',g,(Tstart+dt*n-u)');     % par spline entre chaque point existant
          for q=1:4
                v=W(q,:).*H';
                HH(q,n+1)=0;
                for i=1:4
          HH(q,n+1)=HH(q,n+1)+du*(v(i)+v(i+1))/2;
                end
          end
        end
        hN=[];
        hN(1:2,1)=[HH(3,1);HH(4,1)];
        for n=1:M-1
                hN=[hN; HH(1,n)+HH(3,n+1); HH(2,n)+HH(4,n+1)];
        end
        hN(2*M-1:2*M,1)=[HH(1,M); HH(2,M)];
  end
  Sig=[eye(2*M)];
  er=100;
  K0=ones(2*M,1);
  d=2*M;
  n=0;
  while (er>1e-6 & n<d),            % Itérations jusqu'à convergence du gain K
    if O==1
          Sig=[Sig(1,:)+dt*Sig(2,:);zeros(1,d);Sig(1:d-2,:)];
          Sig=[Sig(:,1)+dt*Sig(:,2) zeros(d,1) Sig(:,1:d-2)];
          Sig(1:2,1:2)=Sig(1:2,1:2)+B*[1/4 1/2;1/2 1];
        else
          Sig=[Sig(1,:)+dt*Sig(2,:);Sig(2,:);Sig(1:d-2,:)];
```

```
            Sig=[Sig(:,1)+dt*Sig(:,2) Sig(:,2) Sig(:,1:d-2)];
            Sig(1:2,1:2)=Sig(1:2,1:2)+B*[dt^4/4 dt^3/2;dt^3/2 dt^2];
        end
        Kt=hN'*Sig;                         % Équations de calcul du gain de Kalman
        Sig=Sig-1/(1+Kt*hN)*Kt'*Kt;
        K=Sig*hN;
        er=norm(K-K0)/norm(K0);
        K0=K;
        n=n+1;
    end
end
V=zeros(2*M,1);          % Calcul de la reconstitution
x=[];
dx=[];
for n=1:N-1
    if O==1
        V=[V(1)+dt*V(2);0;V(1:2*M-2)];      % Matrice phi pour ordre 1
    else
        V=[V(1)+dt*V(2);V(2);V(1:2*M-2)];   % Matrice phi pour ordre 2
    end
    V=V+K*(y(n)-hN'*V);                     % Équation de Kalman
    if c<1                                  % Application  de la contrainte de positivité si demandée
        Vv=reshape(V,2,M);                  % sur les éléments pairs de V
        Vv(1,:)=Vv(1,:).*((Vv(1,:)<0)*c+(Vv(1,:)>=0));
        Vv(2,1)=0;
        for i=2:M-1
            Vv(2,i)=(Vv(1,i-1)-Vv(1,i+1))/(2*dt);      %Correction des dérivées
        end
        V=reshape(Vv,2*M,1);
    end
    if n>M-2                        %Traitement de la translation de l'axe horizontal
        x=[x;V(2*M-1)];
        dx=[dx;V(2*M)];
    end
end
Vv=reshape(V,2,M);                  % Fin du traitement de la translation de l'axe horizontal
x=[x; fliplr(Vv(1,1:M-1))'];
dx=[dx; fliplr(Vv(2,1:M-1))'];
if c<1                              % Contrainte de positivité finale et dure
    for i=1:N
        if (x(i)<0)
            x(i)=0;
        end
    end
end
```

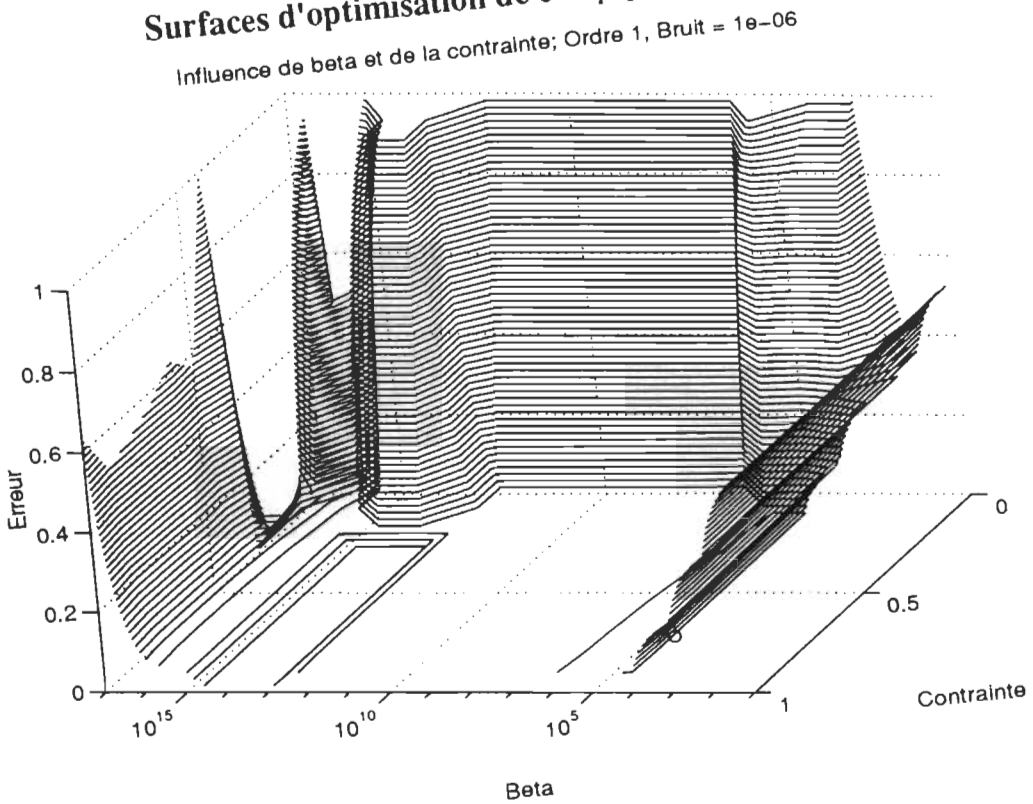
Programme DSP56001

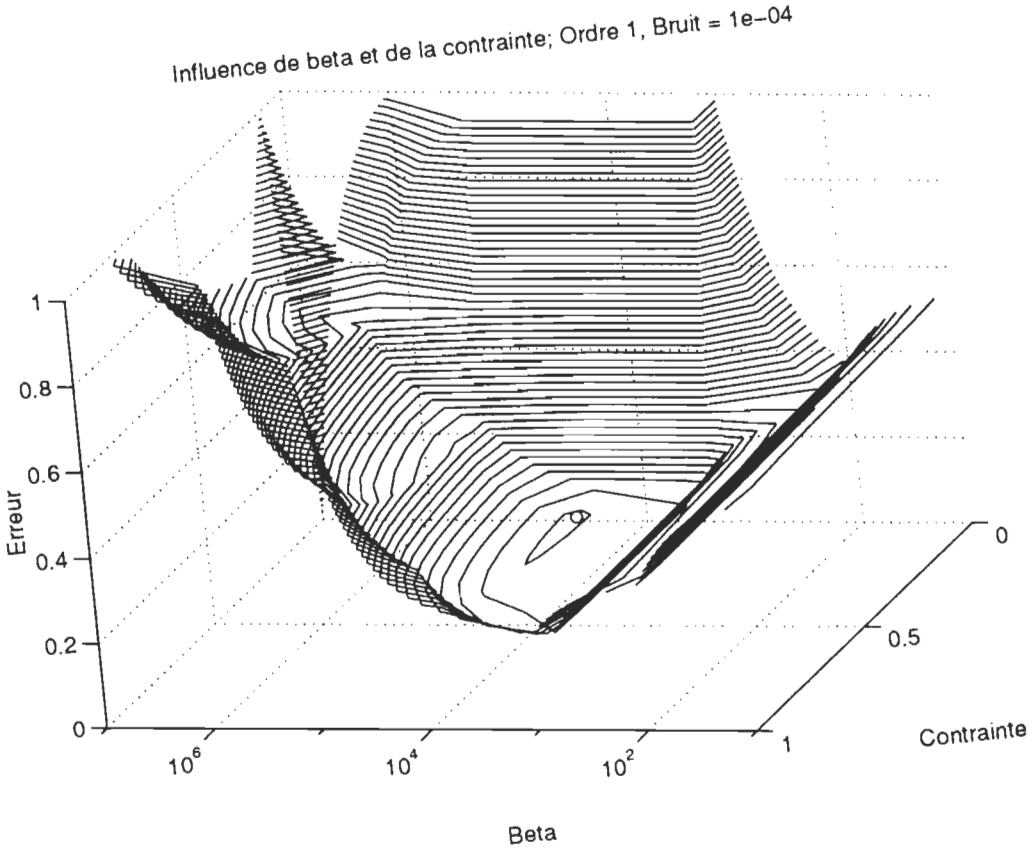
```
N       EQU   64        ; Nombre de points definissant x
        ORG   X:$0
V       DS    2*N       ; Vecteur V
yb      DS    N         ; Vecteur y bruité
        ORG   Y:$0
h       DS    N         ; Vecteur h
K       DS    N         ; Vecteur de gain K
dt      DS    1         ; Pas d'échantillonnage
ntr     DS    1         ; Nombre de translations (divisions par deux) subies par K pour la
                        ; normalisation
c       DS    1         ; Paramètre de contrainte de positivité

        ORG   P:$40
INIT    move  #yb,r1    ; Initialisation des pointeurs et modulos
        move  #V,r0
        move  #(2*N-1),m0
        move  #2,n0
        move  #h,r4
        move  #(N-1),m4
        move  #ntr,r6
        move  #dt,r7
        move  #c,r2
        clr   a           ; R.A.Z. du vecteur V
        rep   #(2*N)
        move  a,X:(r0)+

DEBUT   do    #N-1,FIN          ; Calcul de la reconstitution
        move  #(V+2*N-1),r0
        move  #K,r5
        move  (r0)-n0
TRANS   do    #(2*N-2),F_TRANS  ; Translation de V pour phi
        move  X:(r0)+n0,x0
        move  x0,X:(r0)-n0
        move  (r0)-
F_TRANS nop
        move  (r0)+                       ; Application des équations de phi ordre 1
        move  X:(r0)+,a     Y:(r7),y0
        move  X:(r0)-,x0
        macr   x0,y0,a
        clr    a              a,X:(r0)+
        move  a,X:(r0)-
        move  X:(r0)+,x0    Y:(r4)+,y0                ; Calcul de h*V
        rep    #N
        mac   x0,y0,a       X:(r0)+,x0    Y:(r4)+,y0
        move  #V,r0
        neg    a              X:(r1)+,x0               ; Calcul de I (innovation)
        add   x0,a            (r4)-
        rnd    a
        move  a,x0            Y:(r5)+,y0               ; Calcul de V
MAJ_V   do    #N/2,F_MAJ_V
        mpy   x0,y0,a       X:(r0),y0     ; Pour les éléments pairs (valeurs de x)
```

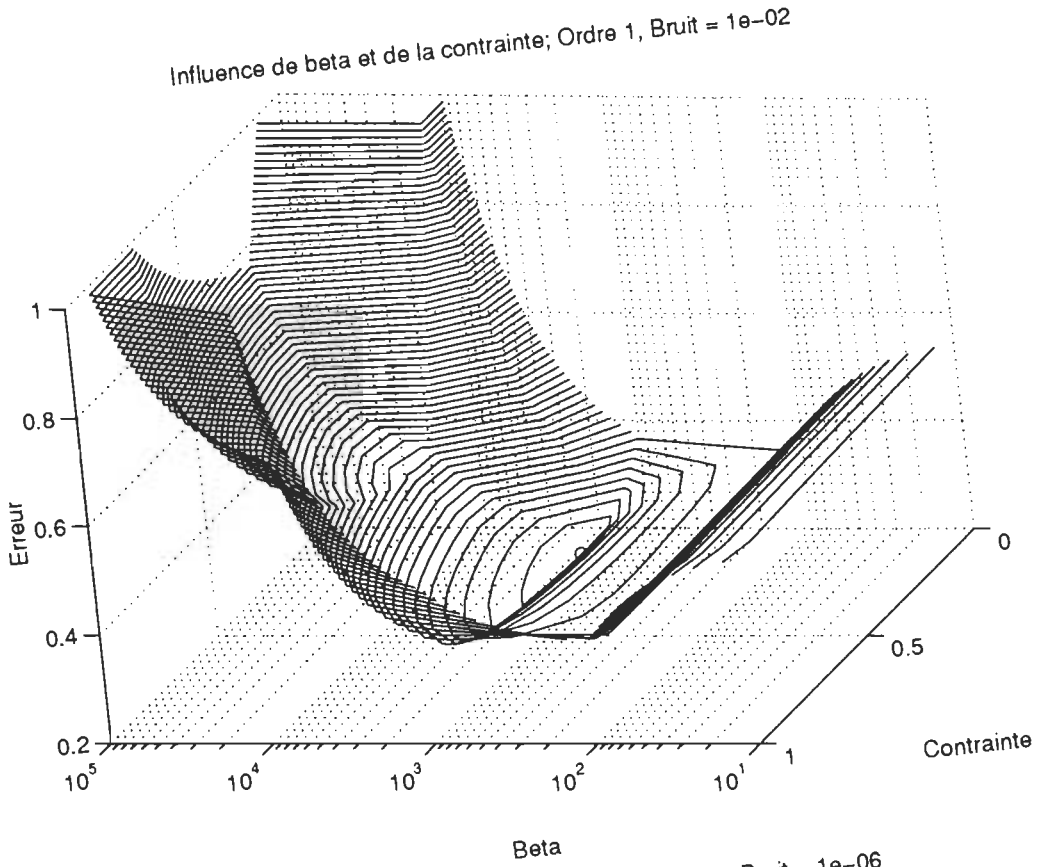
```
        rep     Y:(r6)                          ; Dénormalisation de K
        asl     a
        add     y0,a        Y:(r5)+,y0          ; Calcul de V=V+K I
        rnd     a
        jpl     POSIT                           ; Contrainte de positivité
        move    a,x1        Y:(r2),y1
        mpyr    y1,x1,a
POSIT   move    a,X:(r0)+
        mpy     x0,y0,a     X:(r0),y0   ; Pour les éléments impairs (dérivées de x):
        rep     Y:(r6)                          ; Dénormalisation de K
        asl     a
        add     y0,a        Y:(r5)+,y0          ; Calcul de V=V+K I
        rnd     a
        move    a,X:(r0)+
F_MAJ_V nop
FIN     nop
        move    #V,r0
CONTR   do      #N,F_CONTR              ; Mise à zero de V négatif
        move    x:(r0),a
        rnd     a
        jpl     POSITIF
        move    a,x1
        mpyr    y1,x1,a
POSITIF move    a,X:(r0)+n0
F_CONTR move    (r1)+
        nop
        nop
        END
```

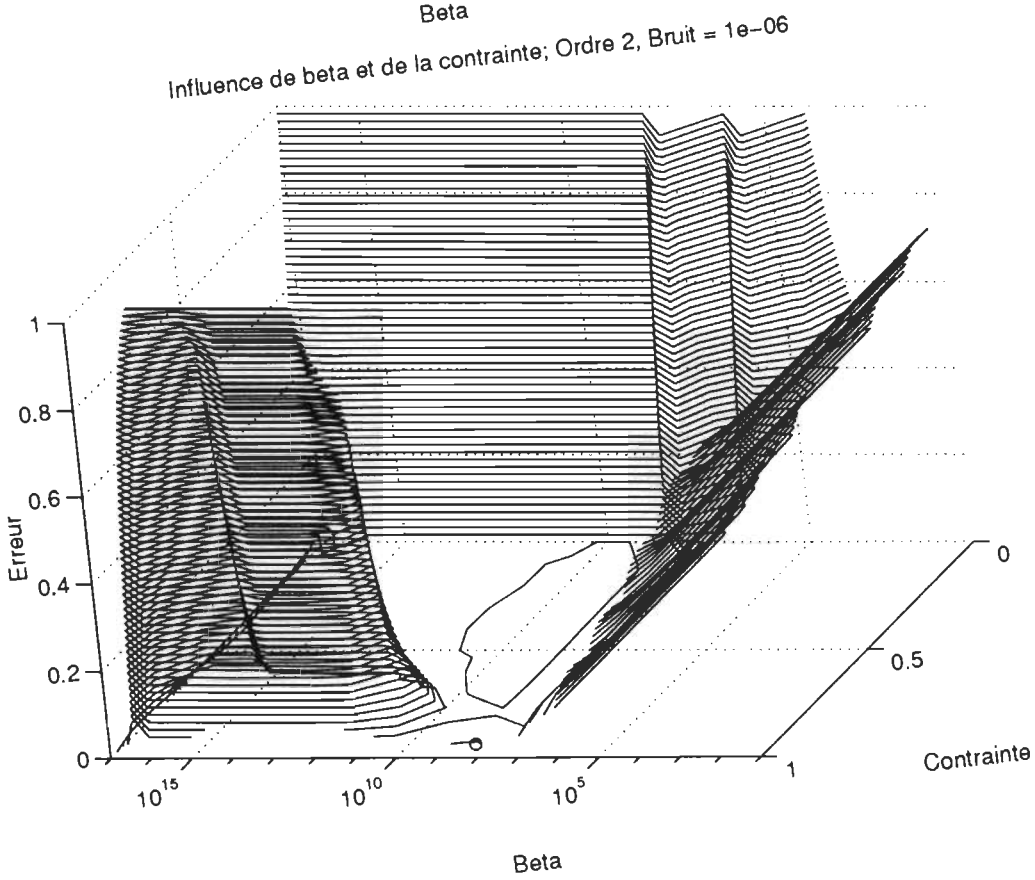
□

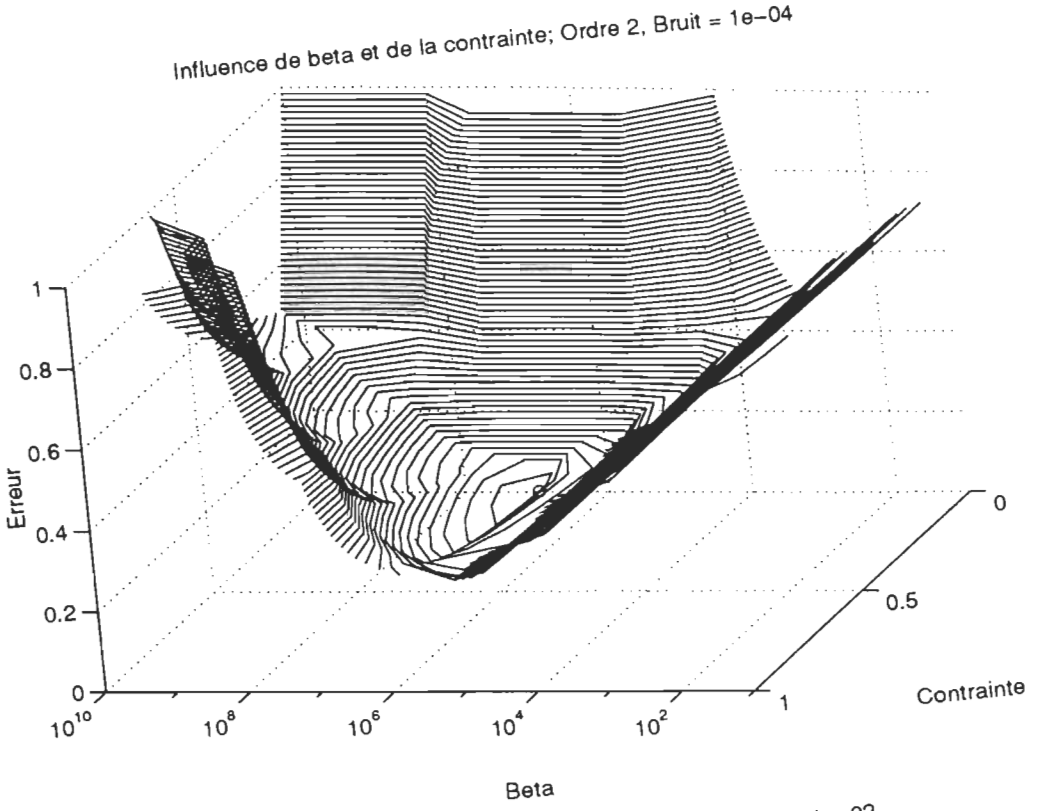
Surfaces d'optimisation de c et β pour Crilly

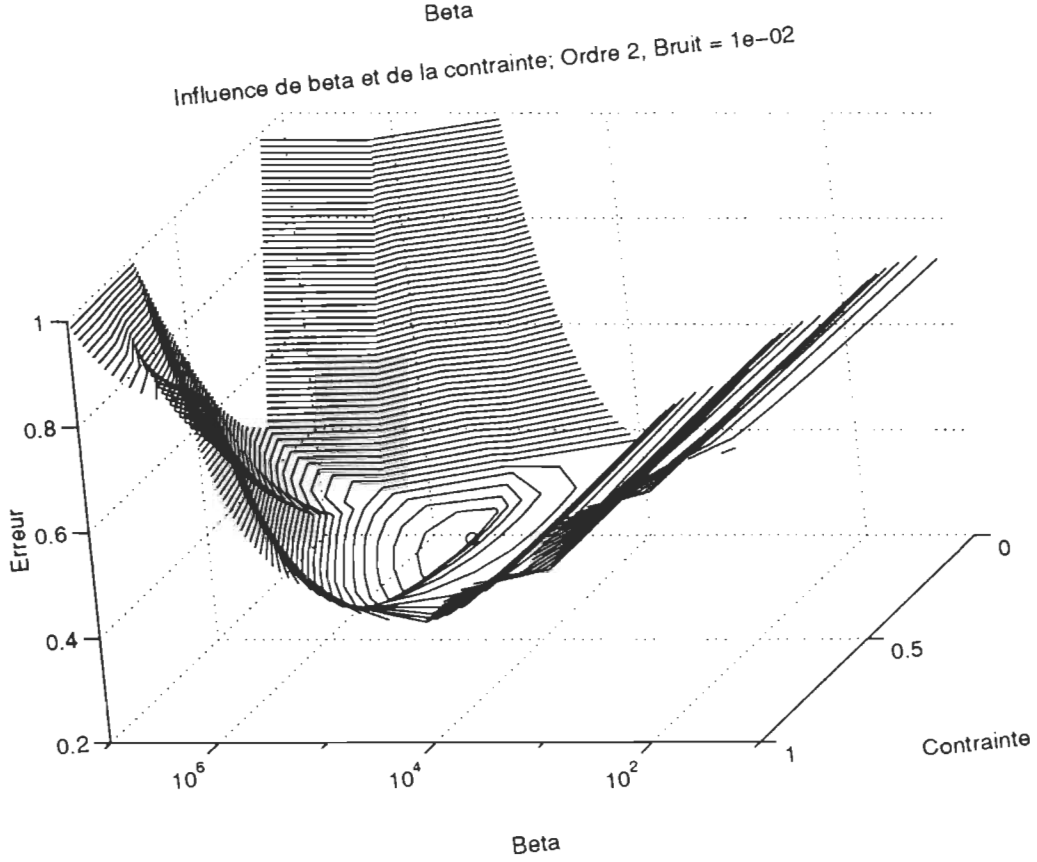





Influence de beta et de la contrainte; Ordre 1, Bruit = 1e−02
Erreur
1
0.8
0.6
0.4
0.2
10^5
10^4
10^3
10^2
10^1
Beta
1
0.5
0
Contrainte


Influence de beta et de la contrainte; Ordre 2, Bruit = 1e−06
Erreur
1
0.8
0.6
0.4
0.2
0
10^{15}
10^{10}
10^5
Beta
1
0.5
0
Contrainte


Influence de beta et de la contrainte; Ordre 2, Bruit = 1e−04
Erreur
1
0.8
0.6
0.4
0.2
0
10^{10}
10^{8}
10^{6}
10^{4}
10^{2}
Beta
1
0.5
0
Contrainte


Influence de beta et de la contrainte; Ordre 2, Bruit = 1e−02
Erreur
1
0.8
0.6
0.4
0.2
10^{6}
10^{4}
10^{2}
Beta
1
0.5
0
Contrainte

Conference Proceedings

IMTC/94

1994 IEEE Instrumentation and Measurement Technology Conference

10th Anniversary

IMTC/94

Advanced Technologies in I & M

Grand Hotel Hamamatsu
Hamamatsu, JAPAN
May 10-12, 1994

浜松

Vol. 3

IEEE Catalog No.94CH3424-9
Library of Congress No.94-75408

Sponsored by
IEEE Instrumentation and Measurement Society
The Society of Instrument and Control Engineers, Japan

DSP-BASED CORRECTION OF SPECTROGRAMS USING CUBIC SPLINES AND KALMAN FILTERING

Pierre BROUARD, Roman Z. MORAWSKI* (Member IEEE) and Andrzej BARWICZ (Member IEEE)

Département d'ingénierie, Université du Québec à Trois-Rivières,
C.P. 500 Trois-Rivières, Québec, G9A 5H7 Canada

*Institute of Radioelectronics, Warsaw University of Technology,
ul. Nowowiejska 15/19, 00-665 Warsaw, Poland

Abstract -- Raw spectrograms are subject to systematic errors of an instrumental type that may be reduced provided a mathematical model of the instrumental imperfections is identified. It is assumed in the paper that this model has the form of an integral, convolution-type equation of the first kind whose kernel may be causal or not. The correction of the spectrograms consists in numerically solving this equation on the basis of the noisy data acquired by a spectrometer. An algorithm of correction is proposed which is based on the approximation of the solution with a spline function whose parameters are determined by means of a recursive Kalman-filter-based algorithm with a non-negativity constraint imposed on the set of feasible solutions. It is shown, using spectrophotometric data, that an improvement in the resolution of the spectrometer can be attained.

I. INTRODUCTION

Spectrogram correction is a particular case of the problem of measurand reconstruction (restoration). This problem arises in almost every branch of engineering or applied physics, especially in metrological applications such as the digital correction of dynamic errors arising in measurement channels, the interpretation of seismic signals or improving the resolution in chromatrography or spectrometry [1]. In this latter domain, the measurements of the absorption spectrum of a sample under study are subject to systematic errors of an instrumental type, due to the imperfections of a spectrophotometer. These errors imply distortion of the essential features of the measured spectrogram (e.g., overlapping of peaks, the appearance of artifacts), making proper interpretation of the real spectrum impossible. To take the total effect of these distortions into account, the convolution model of the recorded spectrum $y(\lambda)$ is most frequently used [2] :

$$y(\lambda) = \int_{-\infty}^{\infty} g(\lambda - \lambda') \cdot x(\lambda') d\lambda' \qquad (1)$$

where $x(\lambda)$ is the undistorted spectrum as it might be recorded by a perfectly resolving spectrophotometer, and $g(\lambda)$ is the unnormalized incoherent optical response function whoses shape resembles a steep gaussoid. Digital correction of spectrograms consists in numerically solving Eq. (1) on the basis of an a-priori identified $g(\lambda)$ and measured samples of $y(\lambda)$ which are inevitably subject to random errors η_n :

$$\tilde{y}_n = y(\lambda_n) + \eta_n \qquad \text{for } n = 0, 1, \ldots, N \qquad (2)$$

This work was supported by the Natural Sciences and Engineering Research Council of Canada (CGP0036573) and by the University of Quebec Hewlett-Packard Canada contributed with equipement support.

Please address the correspondence to :

Andrzej BARWICZ, Département d'ingénierie, Université du Québec à Trois-Rivières, C.P. 500 Trois-Rivières, Québec, G9A 5H7 Canada

This problem of measurand reconstruction is numerically ill-conditioned due to disturbances corrupting the data $\{\tilde{y}_n\}$. The problem must therefore be regularized by constraining the set of admissible solutions. Numerous techniques of regularization have been reported in the literature (see review in [3]), and have been applied to remove the effects of many different types of distortion. Iterative methods are still the most popular techniques in spectrometry - cf. e.g. [4]. However, in the other fields where the methods of measurand reconstruction are applied, like seismology or dynamic calorimetry, much more efficient techniques of regularization have gained popularity, namely varational and parametric methods - cf. [5], [6]. In this paper an algorithm of spectrogram correction is proposed which utilizes both techniques in combination with a non-negativity constraint imposed on the set of admissible solutions. The algorithm is derived using an approximation of the solution with a spline function whose parameters are determined by means of a recursive Kalman-filter-based algorithm and implemented in a digital signal processor (DSP). This approach has four advantages :

- significant flexibility of a spline approximation;
- computational simplicity of the Kalman filter;
- easy incorporation of the non-negativity constraint;
- speed of the DSP.

The proposed method of correction is developed in Section II and III. Next, some details of the DSP implementation are explained in Section IV. In Section V, a real-data-based experiment is reported, while Section VI is devoted to the conclusion.

Throughout the paper, the following notation is used :

λ - a scalar, real variable denoting the wavelength;
$\Delta\lambda$ - the discretization step along the λ-axis;
λ_n - the discretization grid along the λ-axis:
$\lambda_{n+1} - \lambda_n = \Delta\lambda$ for $n = 0,1, \ldots, N$;
$x(\lambda)$ - the scalar, real-valued function of λ, non-zero only for $\lambda \in [\lambda_{min}, \lambda_{max}]$; $\lambda_{min} - \lambda_{max} = M\Delta\lambda$;
$g(\lambda)$ - the scalar, real-valued function of λ, non-zero only for $\lambda \in [\lambda'_{min}, \lambda'_{max}]$; $\lambda'_{min} - \lambda'_{max} = K\Delta\lambda$;
$x^{(k)}(\lambda)$ - the kth derivative of $x(\lambda)$, $k = 1,2, \ldots$;
$y^{(k)}(\lambda)$ - the kth derivative of $y(\lambda)$, $k = 1,2, \ldots$;
$x_m = x(\lambda_m)$ for $m = 0, 1, \ldots, M$;
$g_k = \Delta\lambda \cdot g(\lambda_k)$ for $k = 0, 1, \ldots, K$;
$y_n = y(\lambda_n)$ for $n = 0, 1, \ldots, N$;
$x_m^{(1)} = x^{(1)}(\lambda_m)$ for $m = 0, 1, \ldots, M$;
$x_m^{(2)} = x^{(2)}(\lambda_m)$ for $m = 0, 1, \ldots, M$.

Moreover, the symbol v denotes an exact value of the variable

0-7803-1880-3/94/$4.00 ©1994 IEEE

v, $\bar{v}$ stands for its disturbed value and $\hat{v}$ for its estimate. Vectors and matrices are printed in bold type.

II. MODEL OF MEASUREMENT DATA

The following normalization of the signals $x(\lambda)$, $g(\lambda)$, and $y(\lambda)$ has been made :

$$\begin{aligned} x(\lambda + \lambda_{min}) &\Rightarrow x(\lambda) \text{ for } \lambda \in [0, \lambda_{max} - \lambda_{min}] \\ g(\lambda + \lambda'_{min}) &\Rightarrow g(\lambda) \text{ for } \lambda \in [0, \lambda'_{max} - \lambda'_{min}] \\ y(\lambda + \lambda_{min} - \lambda'_{min}) &\Rightarrow y(\lambda) \text{ for } \lambda \in [0, \lambda_{max} - \lambda_{min} \\ &\qquad + \lambda'_{max} - \lambda'_{min}] \end{aligned}$$

in order to simplify the mathematical modelling of the measurement data $\{y_n\}$. This normalization reduces the integral in Eq. (1) to the form:

$$y(\lambda) = \int_0^{\lambda} g(\lambda - \lambda') \cdot x(\lambda') d\lambda' \tag{3}$$

The proposed approach to spectrogram correction is based on modelling the spectrum $x(\lambda)$ with a cubic spline function [7]

$$\hat{x}(\lambda) = \sum_{k=0}^{3} p_{4n-k} (\lambda - \lambda_n)^k \qquad \text{for} \quad \lambda \in [\lambda_n, \lambda_{n-1}] \tag{4}$$

continuous with its second derivative $\hat{x}^{(2)}(\lambda)$. The parameters p_{4n-k} of this function are related to its knot values x_n and $x_n^{(1)}$ in the following way:

$$p_{4n} = x_n \tag{5}$$

$$p_{4n-1} = x_n^{(1)} \tag{6}$$

$$p_{4n-2} = \frac{1}{\Delta\lambda}\left(3\frac{x_{n-1} - x_n}{\Delta\lambda} - x_{n-1}^{(1)} - 2x_n^{(1)}\right) \tag{7}$$

$$p_{4n-3} = \frac{1}{\Delta\lambda^2}\left(x_{n-1}^{(1)} + x_n^{(1)} - 2\frac{x_{n-1} - x_n}{\Delta\lambda}\right) \tag{8}$$

for n = 0, 1, . . . , N-1. When substituted to Eq. (4), they yield for $\lambda \in [\lambda_n, \lambda_{n-1}]$:

$$\begin{aligned} \hat{x}(\lambda) = {} & w_{00}(\lambda - \lambda_n)x_n + w_{01}(\lambda - \lambda_n)x_n^{(1)} \\ & + w_{10}(\lambda - \lambda_n)x_{n-1} + w_{11}(\lambda - \lambda_n)x_{n-1}^{(1)} \end{aligned} \tag{9}$$

where:

$$w_{00}(\lambda) = 2\frac{\lambda^3}{\Delta\lambda^3} - 3\frac{\lambda^2}{\Delta\lambda^2} + 1 \qquad w_{01}(\lambda) = \frac{\lambda^3}{\Delta\lambda^2} - 2\frac{\lambda^2}{\Delta\lambda} + \lambda$$

$$w_{10}(\lambda) = -2\frac{\lambda^3}{\Delta\lambda^3} + 3\frac{\lambda^2}{\Delta\lambda^2} \qquad w_{11}(\lambda) = \frac{\lambda^3}{\Delta\lambda^2} - \frac{\lambda^2}{\Delta\lambda}$$

Now, the substitution of $x(\lambda')$ in Eq (3) by $\hat{x}(\lambda')$:

$$y_n \cong \int_0^{\lambda_n} g(\lambda_n - \lambda') \cdot \hat{x}(\lambda') d\lambda' \quad \text{for n = 0, 1, ... , N-1} \tag{10}$$

yields the following discrete model of the data:

$$\begin{aligned} y_n \cong \sum_{\upsilon=0}^{n-1} \int_{\lambda_\upsilon}^{\lambda_{\upsilon-1}} & g(\lambda_n - \lambda')[w_{00}(\lambda' - \lambda_\upsilon)x_\upsilon + w_{01}(\lambda' - \lambda_\upsilon)x_\upsilon^{(1)} \\ & + w_{10}(\lambda' - \lambda_\upsilon)x_{\upsilon-1} + w_{11}(\lambda' - \lambda_\upsilon)x_{\upsilon-1}^{(1)}]d\lambda' \end{aligned}$$

$$y_n \cong \sum_{\upsilon=0}^{n-1} \left(h_{n-\upsilon}^{00} x_\upsilon + h_{n-\upsilon}^{01} x_\upsilon^{(1)} + h_{n-\upsilon}^{10} x_{\upsilon-1} + h_{n-\upsilon}^{11} x_{\upsilon-1}^{(1)}\right) \tag{11}$$

where:

$$h_n^{ij} = \int_0^{\Delta\lambda} g(\lambda_n - \lambda') w_{ij}(\lambda') d\lambda' \qquad \text{for i, j = 0, 1} \tag{12}$$

This model may be given the compact form by means of vector notation:

$$y_n = \mathbf{h}_n^T \mathbf{v}_n \tag{13}$$

where:

$$\mathbf{h}_n^T = \left[h_1^{10} \middle| h_1^{11} \middle| h_1^{00} \middle| h_1^{01} \middle| \cdots \middle| h_{n-1}^{00} \middle| h_{n-1}^{01} \middle| h_n^{00} \middle| h_n^{01} \middle| h_n^{10} \middle| h_n^{11} \middle| 0 \middle| \cdots \middle| 0\right]^T$$

$$\mathbf{v}_n = \left[x_n \middle| x_n^{(1)} \middle| x_{n-1} \middle| x_{n-1}^{(1)} \middle| \cdots \middle| x_1 \middle| x_1^{(1)} \middle| x_0 \middle| x_0^{(1)} \middle| 0 \middle| \cdots \middle| 0\right]^T$$

with $h_k^0 = h_k^{00} + h_{k-1}^{10}$, $h_k^1 = h_k^{01} + h_{k-1}^{11}$ for k = 0, 1, ..., K,

and $h_k^{ij} = 0$ for k > K.

In order to make possible the application of the Kalman filter for estimation of $\{x_n\}$, one has to assume that this sequence may be adequately modelled by a discrete stochastic process. Since $x(\lambda)$ has the form of a train of more or less symmetrical peaks, one may suppose that the average value of the derivative $x^{(1)}(\lambda)$ is close to zero, and - as a consequence - assume the following model of the sequence $\{x_n\}$:

$$x_{n-1} = x_n + \Delta\lambda u_n \tag{14}$$

where u_n are realizations of zero-mean random variables $\underline{u}_n$. Thus, taking into account this equation and the structure of the state vector v_n, one may give the model of the measurement data $\{\bar{y}_n\}$ the form of state equations:

$$\mathbf{v}_{n-1} = \Phi \mathbf{v}_n + \mathbf{b} u_n \tag{15}$$

$$y_n = \mathbf{h}_n^T \mathbf{v}_n + \eta_n \tag{16}$$

where:

$$\Phi = \begin{bmatrix} 1 & \Delta\lambda & 0 & \cdots & 0 \\ 0 & 1 & 0 & & \\ 1 & 0 & 0 & \ddots & \vdots \\ 0 & 1 & 0 & & \\ 0 & 0 & 1 & & \\ \vdots & \ddots & & \ddots\ 0\ 0\ 0 & \\ 0 & 0 & \cdots\ 0 & 1\ 0\ 0 & \end{bmatrix} \qquad \mathbf{b} = \begin{bmatrix} \Delta\lambda^2/2 \\ 1 \\ 0 \\ \\ \vdots \\ \\ 0 \end{bmatrix} \tag{17),(18}$$

$$\dim(\Phi) = 2K \times 2K \qquad \dim(\mathbf{b}) = 2K$$

III. METHOD OF CORRECTION

The proposed method of correction is based on the following assumptions concerning the sequences $\{u_n\}$ and $\{\eta_n\}$:

- u_n are realizations of identical, zero-mean, random variables $\underline{u}_n$, and an a-priori estimate $\tilde{\sigma}_u^2$ of their variance σ_u^2 is available;
- η_n are realizations of identical, zero-mean, random variables $\underline{\eta}_n$, and an a-priori estimate $\tilde{\sigma}_\eta^2$ of their variance σ_η^2 is available;
- the variables $\underline{u}_n$ and $\underline{\eta}_n$ are uncorrelated, i.e.: $E[\underline{u}_n\underline{\eta}_n] = 0$, $E[\underline{u}_n\underline{u}_m] = \sigma_u^2\delta_{nm}$, and $E[\underline{\eta}_n\underline{\eta}_m] = \sigma_\eta^2\delta_{nm}$.

Under the above-formulated assumptions, the Kalman filter [8] may be directly applied for estimating the state vector $\mathbf{v}_n$ on the basis of the measurement data $\{\tilde{y}_n\}$ and pre-estimates $\tilde{\sigma}_u^2$ and $\tilde{\sigma}_\eta^2$:

$$\Sigma_{n+1-} = \Phi\Sigma_n\Phi^T + \beta\mathbf{b}\mathbf{b}^T \tag{19}$$

$$\Sigma_{n+1} = \Sigma_{n+1-} - \frac{1}{1+\mathbf{h}_n^T\Sigma_{n+1-}\mathbf{h}_n}\Sigma_{n+1-}\mathbf{h}_n\mathbf{h}_n^T\Sigma_{n+1-} \tag{20}$$

$$\mathbf{k}_{n+1} = \Sigma_{n+1}\mathbf{h}_n \tag{21}$$

$$\hat{\mathbf{v}}_{n+1} = \Phi\hat{\mathbf{v}}_n + \mathbf{k}_{n+1}(\tilde{y}_n - \mathbf{h}_n^T\Phi\hat{\mathbf{v}}_n) \tag{22}$$

where $\beta = \tilde{\sigma}_u^2 / \tilde{\sigma}_\eta^2$, Σ_{n+1} is the normalized (i.e. divided by $\tilde{\sigma}_\eta^2$) covariance matrix of the prediction error, Σ_{n+1-} is the normalized covariance matrix of the estimation error and $\mathbf{k}_{n+1}$ is the vector of Kalman gain. The result of estimation $\hat{\mathbf{v}}_n$ enables one to find the estimates $\hat{x}_n$ of x_n and - consequently - to derive the reconstructed spectrum according to Eq. (9). This solution is relatively time-consuming because the vector $\mathbf{k}_n$ has to be recalculated for n = 1, 2, ..., N. A possibility of simplification results from the following approximation:

$$\mathbf{h}_n^T\mathbf{v}_n \cong \mathbf{h}^T\mathbf{v}_n \tag{23}$$

$$\text{where } \mathbf{h}_n^T = \left[h_1^{10}\middle|h_1^{11}\middle|h_1^{0}\middle|h_1^{1}\middle|\cdots\middle|h_{K-1}^{0}\middle|h_{K-1}^{1}\middle|h_K^{0}\middle|h_K^{1}\middle|h_K^{\infty}\middle|h_K^{01}\right] \tag{24}$$

This approximation enabels one to use a constant value of $\mathbf{k}_n$ in Eq. (22). In the proposed algorithm, called SplKal, the steady-state value of the Kalman gain has been used:

$$\mathbf{k}_\infty = \lim_{n\to\infty}\mathbf{k}_n \tag{25}$$

which may be computed in advance (since it does not depend on the data $\{\tilde{y}_n\}$) using iteratively the formulas (19) and (20), in order to find an approximation of Σ_∞, and the formula

$$\mathbf{k}_\infty = \Sigma_\infty\mathbf{h} \tag{26}$$

Once the steady-state Kalman gain $\mathbf{k}_\infty$ has been calculated, the one-step filtering algorithm is reduced to Eq. (22) with $\mathbf{k}_{n+1}$ replaced with $\mathbf{k}_\infty$.

Further improvement of the quality of correction occurs if a-priori information about the real spectra, namely about their non-negativity, is incorporated into the algorithm of estimation by truncating the negative values of the estimates $\hat{\mathbf{v}}_n$ after each iteration:

$$\hat{\mathbf{v}}_{n,2\nu+1} = \begin{cases} c\hat{\mathbf{v}}_{n,2\nu+1} & \text{if} \quad \hat{\mathbf{v}}_{n,2\nu+1} \le 0 \\ \hat{\mathbf{v}}_{n,2\nu+1} & \text{if} \quad \hat{\mathbf{v}}_{n,2\nu+1} > 0 \end{cases} \quad \text{for: } \nu = 0, 1, \ldots, N \tag{27}$$

where c is a parameter such that $0 \le c \le 1$.

IV. DSP IMPLEMENTATION

The algorithm was first developed and optimized using the MATLAB software. Next it was implemented in the C language on a UNIX workstation to accelerate the study of its computational efficiency for large numbers of measurement data $\{\tilde{y}_n\}$. On the other hand, in order to identify implementation constraints inherent to real applications, the algorithm was implemented in the digital signal processor DSP56001 (Motorola). Since the single-chip mode of this processor does not satisfy the memory requirements of the algorithm, the expanded mode (two blocks of RAM, 64K & 24 bits cells each) was used. Due to the fractional representation of numbers in the DSP56001, all the variables have to be in the range [-1.0, 0.9999998] when stored in the memory. As a rule, the vectors $\tilde{\mathbf{y}}$, h and $\mathbf{k}_\infty$ contain elements which do not comply with this limitation; consequently, they have to be normalized - first the vector h, and next the Kalman gain $\mathbf{k}_\infty$ using the normalized vector h. Because of the range limitation, the normalization cannot be done by the DSP56001 itself - so pre-treatment and post-treatment of the data must be accomplished by an external processor.

V. EXPERIMENTATION WITH REAL-WORLD DATA

The real-world data was acquired by means of the ANRITSU MV02-Series optical spectrum analyser using two high-precision lasers set to the wavelengths of 1.5415 and 1.5435 nm. The following spectra were recorded:

$\{g_{2,n} / n = 1, \ldots, 46\}$ -one-peak spectrum measured by the analyzer set to the resolution 2 nm (Fig. 1);

$\{g_{5,n} / n = 1, \ldots, 88\}$ -one-peak spectrum measured by the analyzer set to the resolution 5 nm (Fig. 1);

$\{\tilde{y}_{0.1,n} / n = 1, \ldots, 128\}$-two-peak spectrum measured by the analyzer set to the resolution 0.1 nm (Fig. 2);

$\{\tilde{y}_{2,n} / n = 1, \ldots, 128\}$ -two-peak spectrum measured by the analyzer set to the resolution 2 nm (Fig. 2);

$\{\tilde{y}_{5,n} / n = 1, \ldots, 128\}$-two-peak spectrum measured by the analyzer set to the resolution 5 nm (Fig. 2);

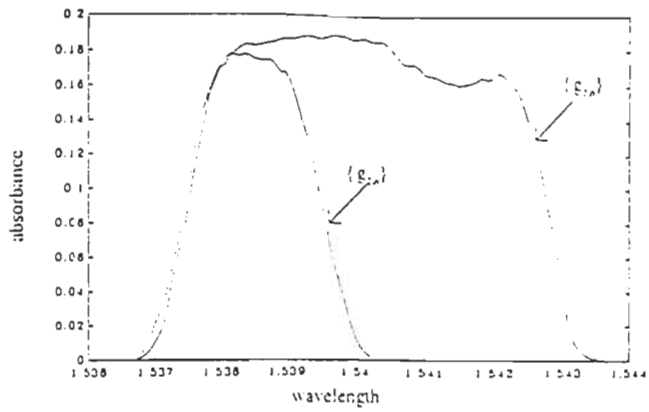


Fig. 1. One-peak spectrum measured by the analyzer set to the resolutions 2 nm and 5 nm

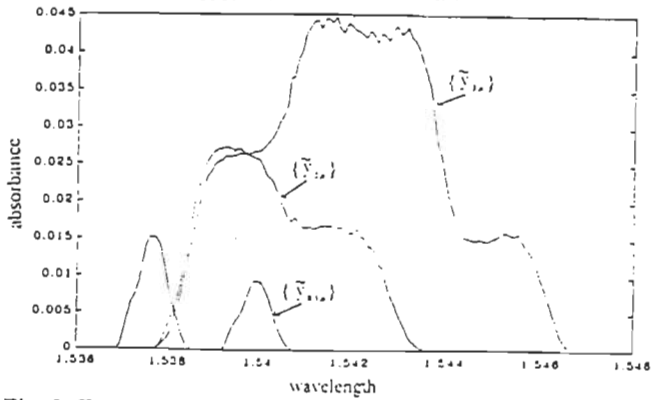


Fig. 2. Two-peak spectrum measured by the analyzer set to the resolution 0.1 nm, 2 nm and 5 nm

The studied algorithm of spectrogram correction, implemented in the DSP56001, was used for processing $\{\tilde{y}_{2,n}\}$ using $\{g_{2,n}\}$ and for processing $\{\tilde{y}_{5,n}\}$ using $\{g_{5,n}\}$; the results are shown in Fig. 3 and Fig. 4.

VI. CONCLUSION

In this paper, we have presented a method for spectrogram correction, based on the smoothing approximation of the spectrum $x(\lambda)$ with a cubic spline whose parameters are determined using the Kalman filter under a non-negativity constraint imposed on the solution. We have demonstrated, using real-world data, that the algorithm implemented in the digital signal processor DSP56001 may be efficiently used for resolution enhancement in spectrometry. This means that a non-expensive spectrometer supported by the studied algorithm implemented in the digital signal processor can yield measurement accuracy comparable with that of an expensive one. An important advantage of the proposed method is its low computational complexity: since the size of the matrices and vectors involved in it is constant (2K) and independent of the number of samples N, for large numbers of samples this method requires much less computation time than, for example, the iterative algorithms [4].

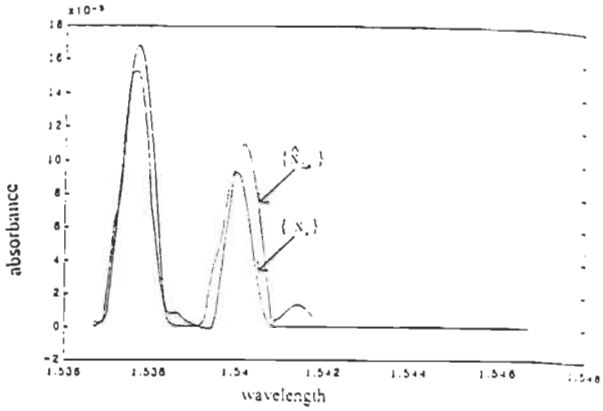


Fig. 3: Result of correction $\{\hat{x}_{2,n}\}$ obtained for $\{\tilde{y}_{2,n}\}$ and compared with $\{x_n\} \equiv \{\tilde{y}_{0.1,n}\}$; parameters of the algorithm: M=128, K=80, β=1, c=0.7.

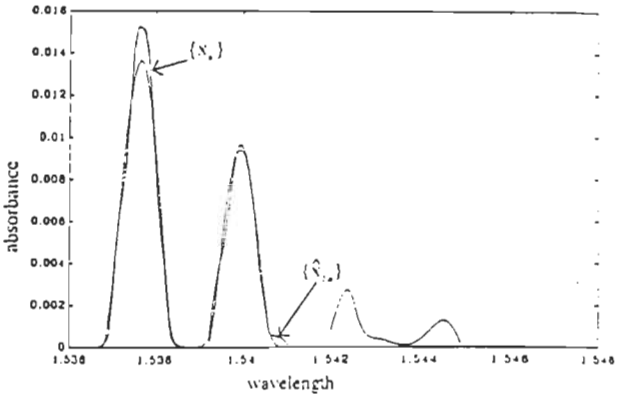


Fig. 4: Result of correction $\{\hat{x}_{5,n}\}$ obtained for $\{\tilde{y}_{5,n}\}$ and compared with $\{x_n\} \equiv \{\tilde{y}_{0.1,n}\}$; parameters of the algorithm: M=128, K=100, β=1, c=0.7.

REFERENCES

[1] R. Z. Morawski, "Metody odtwarzania sygnlow pomiarowych" (Methods of Measurement Signal Restoration), Metrology and Measurement Systems, Monograph No. 1, Warsaw 1989.

[2] P. A. Jansson, Deconvolution with Applications in Spectroscopy, New York: Academic Press Inc., 1984, p.50.

[3] G. Demoment, "Déconvolution des signaux" (Deconvolution of signals), Rapport interne L2S 20/84 du Lab. des Signaux et Systèmes, CNRS/ESE, France 1985.

[4] P. B. Crilly, "A Quantitative Evaluation of Various Iterative Deconvolution Algorithms", IEEE Trans. on Instr. & Meas., Vol. 40, No. 3, June 1991, pp. 558-562.

[5] M. Ben Slima, R. Z. Morawski, A. Barwicz, "Spline-Based Variational Method with Constraints for Spectrophotometric Data Correction", IEEE Trans. on Instr. & Meas., Vol. 41, No. 6, December 1992, pp.786-790.

[6] N. Saito, "Superesolution of Noisy Band-Limited Data by Data Adaptive Regularization and its Application to Seismic Trace Inversion", Proc. IEEE ICASSP'90 (New Mexico, April 3-6, 1990), pp. 1237-1240.

[7] L. L. Schumacher, Spline functions -Basic Theory, New York: Y. Wiley, 1981.

DSP56000/1 FEATURES

- 10.25 Million Instructions Per Second (MIPS)

- Single Cycle, non-pipelined Data ALU
  - 24 x 24 → 56 Bit Parallel Multiply/Accumulate
  - 10 Data Registers
  - 2 Data Bus Shifter/Limiters

- DSP Oriented Address ALU
  - 24 Address Registers
  - Dual Modulo Arithmetic Units
  - Linear, Modulo, and Bit Reversed Address Generation

- Advanced Program Controller
  - 15 Level Hardware Stack
  - Nested Hardware DO Loops
  - No Overhead Auto-Return Interrupts

- Highly Orthogonal Instruction Set
  - 62 Instruction Types
  - Makes Pipeline Invisible
  - Suitable for High Level Language (HLL) Compilers

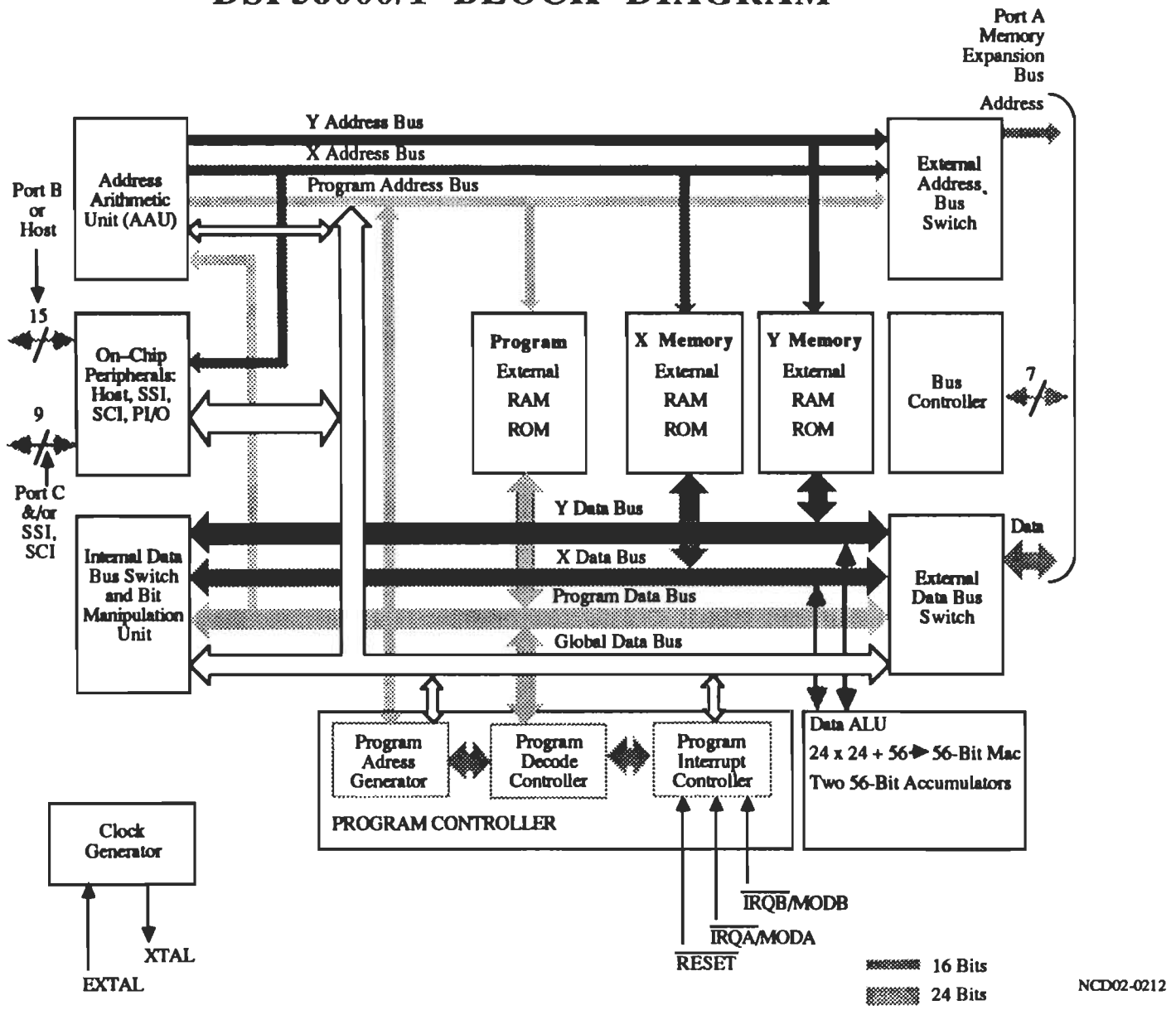
- Multiple Buses
  - 4 Data Buses
  - 3 Address Buses
  - Programmable Off-Chip Access Times

- On-Chip Memory
  - Two Independent 256 x 24-Bit Data RAMs
  - Two Independent 256 x 24-Bit Data ROMs
  - 3.75K x 24-Bit Program ROM (DSP56000 Only)
  - 512 x 24-Bit Program RAM (DSP56001 Only)

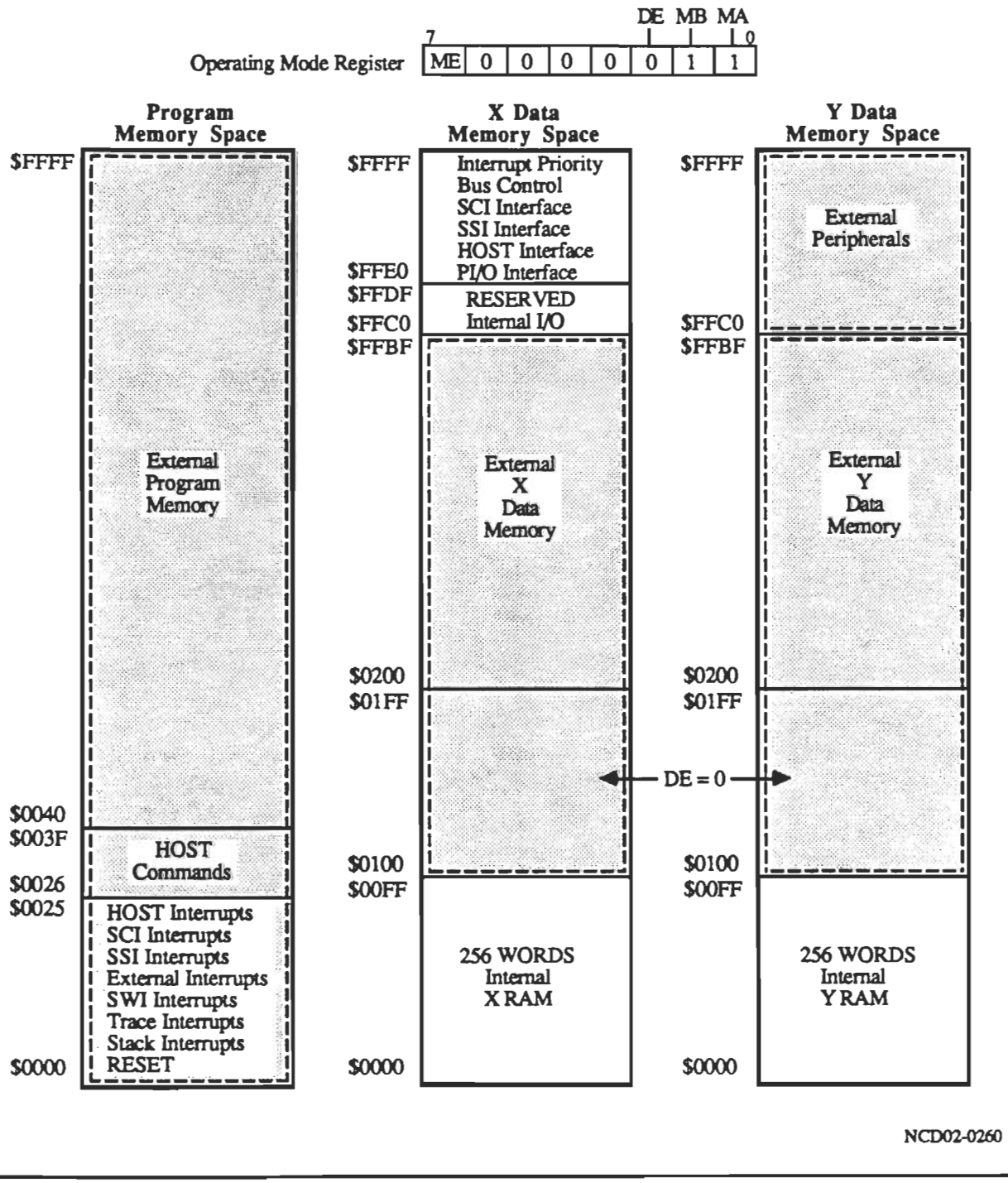
- On-Chip Peripherals
  - 24 Programmable I/O Port Pins
    or a combination of I/O Port Pins and
  - 8-bit Parallel Host MPU/DMA Interface
  - Serial Communication Port with Baud Rate Generator
  - Synchronous Serial (Codec) Port with Clock Generator

NCD02-0112

DSP56000/1 BLOCK DIAGRAM



DSP56001 MEMORY MAP
MODE 3 - DEVELOPMENT MODE
EXTERNAL PROGRAM MEMORY



NCD02-0260

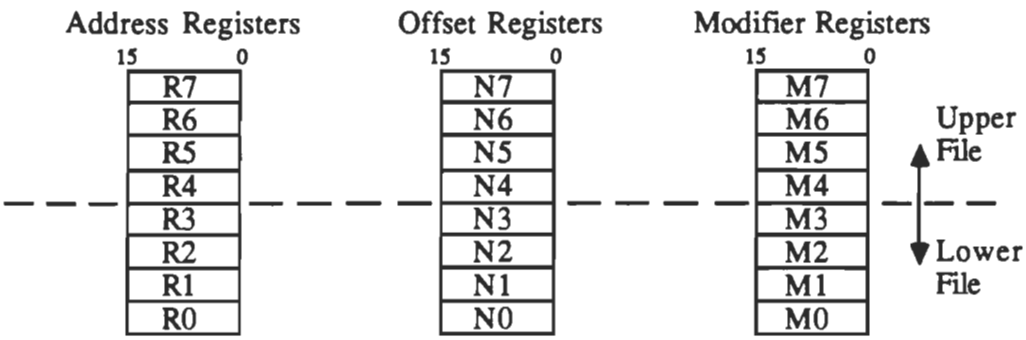
DSP56000/1 PROGRAMMING MODEL

Data ALU

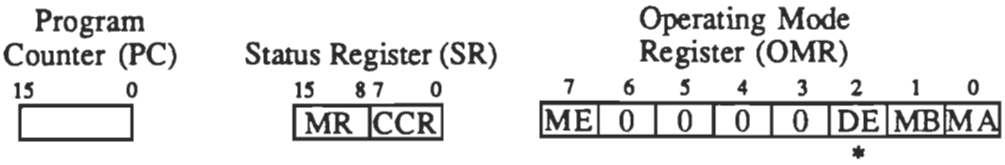
Input Registers

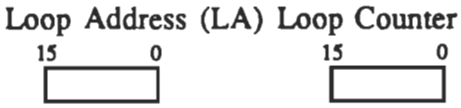
Accumulator Registers

Address ALU

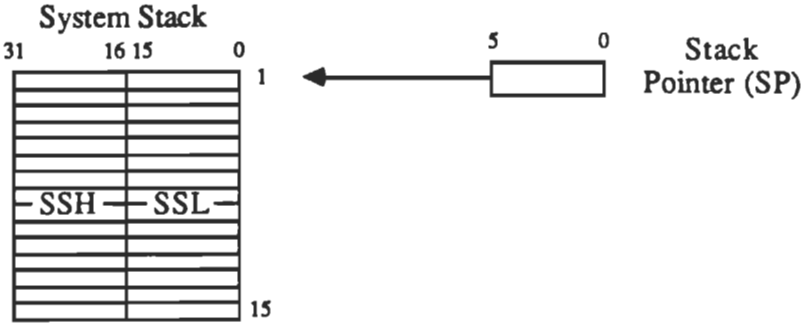


Program Controller

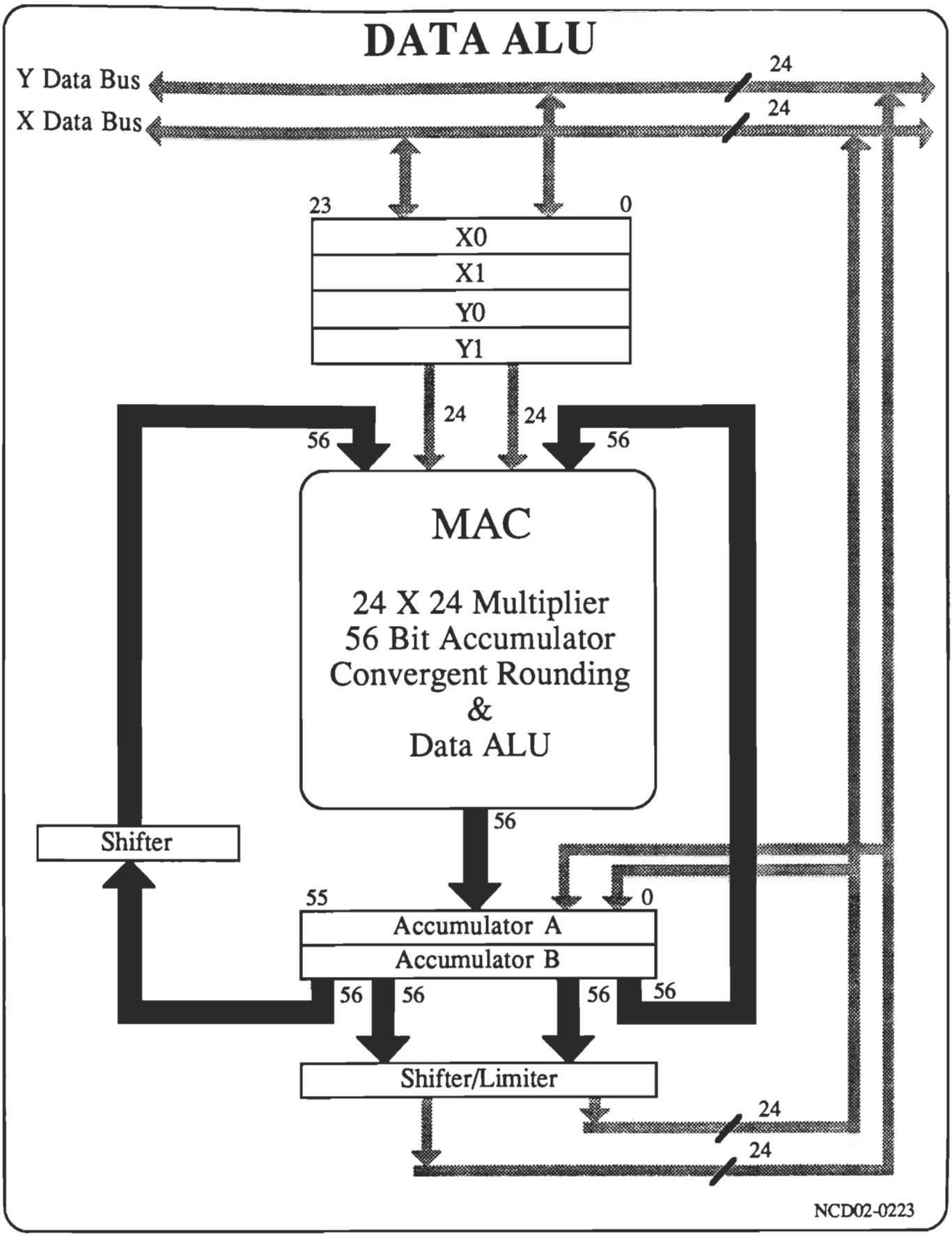




* DE Bit is only on the DSP56001



NCD02-0213


DATA ALU
Y Data Bus
X Data Bus
24
24
23
0
X0
X1
Y0
Y1
24
24
56
56
MAC
24 X 24 Multiplier
56 Bit Accumulator
Convergent Rounding
&
Data ALU
56
Shifter
55
0
Accumulator A
Accumulator B
56
56
56
56
Shifter/Limiter
24
24
NCD02-0223